

**IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF GEORGIA
ATLANTA DIVISION**

**DONNA CURLING, ET AL.,
Plaintiffs,**

v.

**BRAD RAFFENSPERGER, ET AL.,
Defendants.**

Civil Action No. 1:17-CV-2989-AT

**DECLARATION OF ANDREW W. APPEL
IN SUPPORT OF MOTION FOR PRELIMINARY INJUNCTION**

ANDREW W. APPEL, declares, under penalty of perjury, pursuant to 28 U.S.C. § 1746, that the following is true and correct:

1. My name is Andrew W. Appel.
2. I am the Eugene Higgins Professor of Computer Science at Princeton University, where I have been on the faculty since 1986 and served as Department Chair from 2009-2015. I have also served as Director of Undergraduate Studies, Director of Graduate Studies, and Associate Chair in that department. I have served as Editor in Chief of ACM Transactions on Programming Languages and Systems, the leading journal in my field. In 1998 I was elected a Fellow of the

Association for Computing Machinery, the leading scientific and professional society in Computer Science.

3. I previously provided a Declaration in support of the Curling Plaintiffs' Reply in Support of their Motion for Preliminary Injunction on December 13, 2019 (Dkt. No. 681-3). My 2019 Declaration is attached as **Exhibit A**. I have reviewed my 2019 Declaration and my previous findings and analyses remain the same; accordingly, I incorporate by reference my prior Declaration in its entirety, subject to the additional opinions I offer here.

4. My background, qualifications, and professional affiliations were previously identified in my 2019 Declaration and accompanying CV. I have over 40 years' experience in computer science, and 15 years' experience studying voting machines and elections. I am not being compensated for my work related to this matter. I expect that my expenses, if any, will be reimbursed.

5. I previously commented on the Declaration of Juan E. Gilbert, submitted 13 November 2019 in my 2019 Declaration. My opinions regarding the shortcomings of Mr. Gilbert's initial analysis have not changed. In addition, I have read the Supplemental Declaration of Juan E. Gilbert in this case, as well as the Declaration of Jack Cobb, both submitted 26 August 2020.

Gilbert Supplemental Declaration

6. Professor Gilbert’s Supplemental Declaration attempts to address the question of voter verification of votes cast on BMD systems, but his conclusions do not address (nor dispute) the underlying vulnerabilities associated with the use of BMDs as I understand them to be implemented in Georgia. Professor Gilbert does not address any of my prior conclusions regarding the fundamental insecurities with BMDs.

7. In paragraph 7A, Professor Gilbert makes much of the fact that Georgia’s State Election Board (“SEB”) has issued rules intended to require poll workers to remind voters to review their votes before scanning them. Professor Gilbert cites to recently published research from Kortum, Byrne, and Whitmore that suggests reminders to voters to review their ballots.¹

8. But even Professors Kortum and Byrne and Ms. Whitmore acknowledge the limitations of their own study: “it seems that the next logical question is ‘What can be done to get people to take the time to examine their ballots in the first place?’ There are a number of possibilities, all of which would require additional research in order to understand how efficacious they might be.”² (page 16). And they

¹ Philip Kortum, Michael D. Byrne, Julie Whitmore, Voter Verification of BMD Ballots Is a Two-Part Question: Can They? Mostly, They Can. Do They? Mostly, They Don’t (2020) [hereafter Kortum et al.], available at <https://arxiv.org/abs/2003.04997>.

² Kortum et al., 16.

explicitly acknowledge the problem raised by Appel, DeMillo, and Stark,³ when they write, “Of course, if such efforts are successful in getting most or all of the voters to check their ballots, then we must also investigate how to effectively deal with people finding errors and making sure that those are viewed not as human mistakes but as warning signs of a potential malicious agent in a BMD.”⁴ (page 16)

9. That is: even if some voters detect that the BMD printed fraudulent votes onto their ballot, there’s no effective remedy. That is the problem that Kortum, Byrne, and Whitmore say “we must also investigate.”

10. Importantly, Professor Gilbert does not address this more fundamental problem. Even if a handful of voters in a precinct are able to detect an error, there is no way to prove it and such detection provides no basis to invalidate an entire election, even though the election potentially should be invalidated because of vote-altering malware.

11. In paragraph 12 of his Supplemental Declaration, Professor Gilbert also reiterates his belief that risk-limiting audits (“RLAs”) could help detect the presence of malware, but this conclusion ignores the shortcoming of RLAs when applied to BMDs. RLAs depend on a *trustworthy* record of the vote expressed by the voter.

³ Andrew W. Appel, Richard A. DeMillo, & Philip B. Stark, Ballot-marking devices (BMDs) cannot assure the will of the voters (2019) [hereinafter Appel et. al.]. A copy of this research is attached as **Exhibit B** to this Declaration.

⁴ Kortum et al., 16.

When the source of the paper trail is susceptible to hacking, bugs, or other malfunctions, as is the case with BMDs, it is not trustworthy. There simply is no method for checking whether any errors in how BMDs record expressed votes altered election outcomes.⁵

Cobb Declaration

12. Mr. Cobb is an employee of the firm paid by the Secretary of State to conduct testing of the BMD system for “Georgia-specific election criteria.” (Cobb Decl. ¶ 6.) Mr. Cobb does not profess to be an expert in election security or in computer security.

13. In Paragraph 5 of his Declaration, Mr. Cobb mentions some testing performed on the Dominion BMD system now used in Georgia, calling it a “security test.” Mr. Cobb’s firm did not perform this testing, and he does not describe any specifics of the certification and testing performed by SLI Compliance. Mr. Cobb references standards developed by the Pennsylvania Department of State for “penetration testing,” but it does not appear that Mr. Cobb’s firm has ever performed any penetration testing of any elements of the Georgia BMD system. Mr. Cobb does not point to any documentation that would show the scope, limitations, or any other details of penetration testing performed on the Georgia BMD system (before or after its deployment across the state), nor does he identify the results of such testing.

⁵ Appel et al., 3, 8-9.

14. It is well understood in the cybersecurity industry that penetration testing, even when properly conducted, can only demonstrate the presence of security vulnerabilities. Penetration testing serves to exploit a system's weaknesses in order to improve security. However, penetration testing cannot demonstrate that a system is free of vulnerabilities, as there may be numerous avenues of attack that are not explored by even comprehensive penetration testing. In addition, penetration testing cannot necessarily demonstrate whether a system has already been infected. This is why penetration testing is used as only one component of a comprehensive risk assessment system.

15. In paragraphs 7 and 8, Mr. Cobb described the acceptance testing his firm performed for Georgia in August 2019, and the use of a hash value to determine that "the correct software" was installed at the time of acceptance. Such reliance on hash values is severely misplaced. It is well understood in the cybersecurity industry that fraudulent software can easily mimic legitimate software by displaying the same hash code. Thus, the use of hash codes does not provide any assurance that the correct software is installed.

16. Based on my preliminary review of the certification report produced by Mr. Cobb's firm in August 2019,⁶ the testing performed by Pro V&V appears to have

⁶ Pro V&V, Test Report Dominion Voting Systems D-Suite 5.5-A Voting System, Georgia State Certification Testing, (Aug. 7, 2019) *available at* https://sos.ga.gov/admin/uploads/Dominion_Test_Cert_Report.pdf

been limited in scope. The report does not represent itself as a security analysis and the testing does not appear to have included any comprehensive security testing.

The accuracy and acceptance testing Mr. Cobb describes in Paragraphs 6 and 7 of his Declaration are not a substitute for a full security assessment.

17. There is clear consensus in the election security community that there are many layers between “the application software that implements an election function and the transistors inside the computers that ultimately carry out computations.”⁷

Any one of these layers could serve as a vector for attack that could introduce fraudulent vote-counting software. Based on my preliminary view of the findings from Pro V&V, it does not appear that any of these layers underlying Georgia’s BMD system were examined. At a minimum, if Pro V&V were to have set out to assess the security of the BMD system, their examination would have included the BIOS and operating-system layers; it would have included insider threats as well as vulnerabilities to external hackers; it would have included a tool-based static analysis of the software; and it would have included penetration testing with a comprehensive description of which layers were attacked.

18. Therefore, nothing in Mr. Cobb’s declaration or Professor Gilbert’s supplemental declaration alter my previous conclusions: (1) the BMD system used

⁷ National Academies of Sciences, Engineering, and Medicine (“NASEM”), *Securing the Vote: Protecting American Democracy* (2018), pp. 89-90.

in Georgia can be hacked to alter votes systematically before printing them onto the paper ballot; (2) most voters will not notice such alteration; (3) if some voters do notice an alteration, there is no effective remedy that election officials can provide⁸; and (4) any such alteration cannot be corrected by a recount or random audit of the paper ballots, because the fraudulent votes are already printed onto those ballots.

I declare under penalty of the perjury laws of the State of Georgia and the United States that the foregoing is true and correct and that this declaration was executed 9/1/20 in Princeton, NJ.



ANDREW W. APPEL

⁸ Voters who notice an error of this kind can be given the opportunity to recast *their own* ballots, but voters who do not notice (which will be the majority of voters even in the most optimistic scenarios in the scientific literature) cannot correct their “hacked” ballot. Therefore the remedy is not *effective*.

EXHIBIT A

**IN THE UNITED STATES DISTRICT COURT
FOR THE NORTHERN DISTRICT OF GEORGIA
ATLANTA DIVISION**

**DONNA CURLING, ET AL.,
Plaintiffs,**

v.

**BRAD RAFFENSPERGER, ET AL.,
Defendants.**

Civil Action No. 1:17-CV-2989-AT

**DECLARATION OF ANDREW W. APPEL
IN SUPPORT OF MOTION FOR PRELIMINARY INJUNCTION**

ANDREW W. APPEL, declares, under penalty of perjury, pursuant to 28 U.S.C. § 1746, that the following is true and correct:

1. My name is Andrew W. Appel.
2. My background, qualifications, and professional affiliations are set forth in my curriculum vitae, which is attached as Exhibit A. I have over 40 years' experience in computer science, and 15 years' experience studying voting machines and elections.
3. I am the Eugene Higgins Professor of Computer Science at Princeton University, where I have been on the faculty since 1986 and served as Department Chair from 2009-2015. I have also served as Director of Undergraduate Studies, Director of Graduate Studies, and Associate Chair in that department. I have

served as Editor in Chief of ACM Transactions on Programming Languages and Systems, the leading journal in my field. In 1998 I was elected a Fellow of the Association for Computing Machinery, the leading scientific and professional society in Computer Science.

4. I received an A.B. (1981) from Princeton University *summa cum laude* in Physics, and a PhD (1985) from Carnegie Mellon University in Computer Science.

5. I have taught undergraduate and graduate courses at Princeton University in programming, programming languages, software engineering, election machinery, software verification, and formal methods.

6. I have testified on election technology before the U.S. House of Representatives (subcommittee on information technology, 2016), the New Jersey legislature (several committees, on several occasions 2005-2018), the Superior Court of New Jersey (Mercer County, 2009; Cumberland County, 2011), the New York State Board of Elections (2019), the Freeholders of Mercer County (2017 and 2019) and Essex County (2019).

7. I have published over 100 scientific articles and books, including many papers on computer security and several papers on voting machines, election technology, and election audits.

8. I have served as a peer-review referee for the Usenix Electronic Voting Technology workshop.

9. I am not being compensated for my work related to this matter. I expect that my expenses, if any, will be reimbursed.

10. I have read the Declaration of Juan E. Gilbert in this case, dated 13 November 2019. His Declaration is remarkable for what he does *not* say.

11. Between November 2018 and March 2019 I conducted a research collaboration with Professor Rich DeMillo of Georgia Tech and Professor Philip Stark of U.C. Berkeley, leading to the publication of our joint paper, “Ballot Marking Devices (BMDs) cannot assure the will of the voters,” (by Appel/DeMillo/Stark) released in April 2019. Our research analyzes the consequences of an important study by DeMillo, Kadel, and Marks released December 2018 entitled “What Voters are Asked to Verify Affects Ballot Verification: A Quantitative Analysis of Voters' Memories of Their Ballots.”

12. Professor Stark’s Declaration focuses on the scientific results of these two papers. Professor Gilbert does not attempt to rebut the key findings of these papers: BMD-marked ballots are not adequately voter-verified, and thus BMD-for-all-voters elections are not secure.

13. The DeMillo/Kadel/Marks paper describes two different studies, two separate aspects of the same question. (1) Do voters review the ballot-cards produced by BMDs before they insert those cards in the optical scanner? and (2) How much can they remember about what contests were on the ballot?

Measurements of real voters in a real polling place in Tennessee answered question 1 as, “47% of voters are seen *not to look at the paper at all*, and the other 53% look at the paper *for an average of 3.9 seconds, even though there were 18 contests on the ballot.*” Interviews with those same voters outside the polling place showed that the answer to question 2 is, “not very accurately.”

14. In our April 2019 paper we analyze the consequences of Finding 1, that most voters hardly examine the BMD-marked paper ballot at all. Finding 2 was interesting but not consequential to our analysis.

15. Our analysis asks: if few voters examine their BMD-marked paper ballots, then what? Surely *a few* voters will examine their ballot, so that if the BMDs have been hacked to steal 10% of the votes, and 10% of the voters carefully examine their ballots, and half of those voters are not too timid to alert a pollworker when they notice something wrong, then only *1 in 200 voters* will alert a pollworker. You might think, “these voters caught the BMD cheating red-handed, surely there will be consequences!” But our analysis demonstrates that there can be no consequences: the BMD will have succeeded in stealing many votes; election officials cannot invalidate elections just because a few voters claimed their ballot was wrongly marked.

16. Professor Gilbert, in paragraph 63 of his Declaration, calls the DeMillo/Kadel/Marks paper a “flawed” study, and *all* of his criticisms of it

concern Finding 2. He does not address or dispute Finding 1 at all: most voters don't even look at the paper. It is Finding 1 that is most important, and on which we based our further analysis.

17. Some of Professor Gilbert's own very recent research is motivated by exactly these problems that the Appel/DeMillo/Stark paper and the DeMillo/Kadel/Marks paper identified in Ballot-Marking Devices. In April 2019 he publicly proposed a "ballot-marking verification protocol"¹ and in May 2019 proposed a "transparent interactive printing interface for voting."²

18. *Both of Professor Gilbert's new studies are premised on the existence of a real problem with voter verification of BMD-marked ballots.* Professor Gilbert avoids criticizing Finding 1 of the DeMillo/Kadel/Marks paper, and he does not address the Appel/DeMillo/Stark paper at all, even though this result is a central point of Professor Stark's Declaration, and Professor Gilbert does address other points of that Declaration.

19. These two of Professor Gilbert's research projects have not yet produced results that are usable in real elections, but they illustrate that he takes seriously the problem that we identified with BMDs, and that he did not rebut in his declaration.

¹ Ballot Marking Verification Protocol, by Juan E. Gilbert, Ph.D., <http://www.juangilbert.com/BallotMarkingVerificationProtocol.pdf> The document is undated but I first saw it on April 13, 2019.

² Transparent Interactive Printing Interface for Voting, by Juan E. Gilbert, Ph.D., <https://hxr.cise.ufl.edu/PrimeIII/TIPI/TransparentInteractivePrintingInterfaceForVoting.pdf> The document is undated but I first saw a version of it on May 12, 2019.

20. This absence of rebuttal—from an expert demonstrably familiar with the substance of both papers—speaks volumes. There is a real problem with BMD-marked paper ballots: voters don’t inspect them, and if a voter says there’s an error, there’s no way to prove it.

21. In paragraphs 72-75, Professor Gilbert addresses the “Curling Plaintiffs’ Exhibit 4: Paper authored by Appel, DeMillo, and Stark.” In these paragraphs he states that he disagrees with our *policy conclusions* (that voters who can hand-mark an optical-scan paper ballot should be permitted to do so), but he does not say that he disagrees with our *scientific conclusion*: “Risk-limiting audits of a trustworthy paper trail can check whether errors in tabulating the votes *as recorded* altered election outcomes, but there is no way to check whether errors in how BMDs *record* expressed votes altered election outcomes. The outcomes of elections conducted on current BMDs therefore cannot be confirmed by audits.”³

22. In paragraph 59, Professor Gilbert is simply and obviously wrong. He is responding to Professor Stark’s statement that “Bugs, misconfiguration, or malicious hacking can cause the BMD to print something other than the selections the voter made on the touchscreen or accessible interface. Hand-marked paper ballots do not have that vulnerability.” Professor Gilbert writes, “This is simply not true.” But it very simply is true, on the face of it. Bugs, etc. cannot cause a BMD to print wrong

³ This is a direct quotation from the abstract of the paper.

selections on a hand-marked paper ballot. Hand-marked paper ballots do not have that vulnerability.

23. In paragraph 61, Professor Gilbert writes, “I disagree with Dr. Stark that hand-marked paper ballots are ‘strongly software independent.’ ” We can simply look at the definition. Rivest defines⁴: “A voting system is *software independent* if an undetected change or error in its software cannot cause an undetectable change or error in an election outcome.” They define, “A voting system is strongly software-independent if an undetected change or error in its software cannot cause an undetectable change or error in an election outcome, and moreover, a detected change or error in an election outcome (due to change or error in the software) can be corrected without re-running the election.”

24. Hand-marked paper ballots are software independent because no change or error in *software* can affect what the voter marks on the paper, and no change or error in *software* can affect what the human recounters or auditors see on the paper. Professor Gilbert is simply wrong on this point.

25. In paragraph 62 Professor Gilbert points out that badly designed paper ballots can lead to substantial unintended undervoting by voters, as in Broward County,

⁴ Rivest, Ronald L. "On the notion of 'software independence' in voting systems." *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366.1881 (2008): 3759-3767.

Florida. This is true; I’ve written about this myself.⁵ He neglects to mention that badly designed touchscreen ballots can *also* lead to substantial unintended undervoting, as in Sarasota, Florida.⁶ There are Federally recognized guidelines for good ballot design;⁷ whether for hand-marked ballots or for BMD touchscreen layout, election administrators would be wise to follow them.

26. In paragraph 38 Professor Gilbert writes, “In theory, a scanner could be programmed to reject an overvoted ballot...” This is more than just theory, it is the practice in many states (e.g., New York) that use precinct-count optical scan of hand-marked ballots. Voters *are* protected against overvote mistakes; BMDs have no accuracy advantage in this respect.

27. In paragraph 39 Professor Gilbert claims that BMDs have advantages in “Auditability, Recounts, and Voter Intent”, but several of his specific examples are inapposite or simply wrong.

28. In paragraph 39A Professor Gilbert opines that hand-marked paper ballots cannot be audited because some voters might make imperfect marks, but many states can and do successfully perform audits of hand-marked paper ballots (Colorado,

⁵ Florida is the Florida of Ballot-Design Mistakes, by Andrew W. Appel, November 14, 2018. <https://freedom-to-tinker.com/2018/11/14/florida-is-the-florida-of-ballot-design-mistakes/>

⁶ What Happened in Sarasota County? by David Jefferson, December 3, 2008. National Academy of Engineering: <https://www.nae.edu/19582/Bridge/VotingTechnologies/WhatHappenedinSarasotaCounty.aspx>

⁷ Effective Designs for the Administration of Federal Elections, Section 3: Optical scan ballots. U.S. Election Assistance Commission, 2007. <https://www.eac.gov/assets/1/1/Effective%20Designs%20for%20the%20Administration%20of%20Federal%20Elections%20-%20Optical%20Scan%20Ballots.pdf>

California, Ohio, Rhode Island, just to name a few of which I have personal knowledge.)

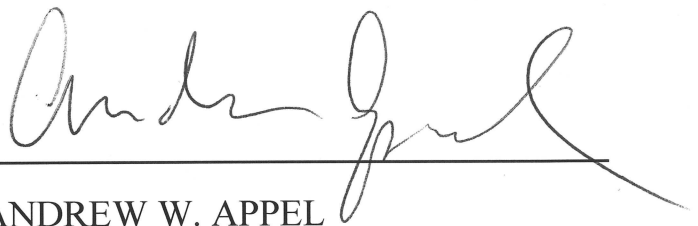
29. In Paragraph 39B he suggests that hand-marked paper ballots cannot be recounted because some voters make imperfect marks, and he specifically names Florida's 2000 Presidential Election and the 2008 Minnesota Senate Race. But Florida's 2000 Presidential Election used punched cards, not optical-scan ballots; punchcards are a grossly inferior technology and have no implication for the auditability of paper ballots. The 2008 Minnesota Senate race was successfully recounted; I wrote an analysis of that case at the time, showing that only a tiny percentage of the paper ballots were ambiguously marked.

30. In Paragraph 39C he writes, "Ambiguous marks cannot occur on a BMD: the voter's intent is clear in the ballot summary..." but *completely* ignores voter-intent problems such as miscalibrated touchscreens and the fact that most voters do not look at the ballot summary.

31. In Paragraph 39F he speculates about future technology; even assuming his speculation someday becomes relevant to what the State purchases and implements, he has neglected to analyze the consequences of the fact that a QR-based audit trail can also be hacked.

32. BMD-marked paper ballots are insecure because: BMDs, like any computers, can be hacked (by alteration of their software program to cheat); if

hacked, they can systematically change votes from what the voter indicated on the touchscreen when printed on the paper ballot; few voters will notice, and those that notice have *only* the mitigation that they might be able to correct their own ballots, not their neighbors; and finally, recounts or audits will see only the fraudulently marked paper. This is the central point of Professor Stark's and Professor Halderman's Declarations; and Professor Gilbert avoids disputing these central facts.

A handwritten signature in black ink, appearing to read "Andrew Appel", written over a horizontal line.

ANDREW W. APPEL

December 13, 2019
Princeton, NJ

EXHIBIT B

EXHIBIT B

Ballot-marking devices (BMDs) cannot assure the will of the voters

Andrew W. Appel[†]
Princeton University

Richard A. DeMillo[†]
Georgia Tech

Philip B. Stark[†]
Univ. of California, Berkeley

April 21, 2019

Abstract

Computers, including all modern voting systems, can be hacked and misprogrammed. The scale and complexity of U.S. elections may require the use of computers to count ballots, but election integrity requires a paper-ballot voting system in which, regardless of how they are initially counted, ballots can be re-counted by hand to check whether election outcomes have been altered by buggy or hacked software. Furthermore, secure voting systems must be able to recover from any errors that might have occurred.

However, paper ballots provide no assurance unless they accurately record the vote as the voter *expresses* it. Voters can express their intent by hand-marking a ballot with a pen, or using a computer called a ballot-marking device (BMD), which generally has a touchscreen and assistive interfaces. Voters can make mistakes in *expressing* their intent in either technology, but only the BMD is *also* subject to systematic error from computer hacking or bugs in the process of recording the vote on paper, after the voter has expressed it. A hacked BMD can print a vote on the paper ballot that differs from what the voter expressed, or can omit a vote that the voter expressed.

It is not easy to check whether BMD output accurately reflects how one voted in every contest. Research shows that most voters do not review paper ballots

[†]Authors are listed alphabetically; they contributed equally to this work.

printed by BMDs, even when clearly instructed to check for errors. Furthermore, most voters who do review their ballots do not check carefully enough to notice errors that would change how their votes were counted. Finally, voters who detect BMD errors before casting their ballots, can correct only their own ballots, not systematic errors, bugs, or hacking. There is no action that a voter can take to demonstrate to election officials that a BMD altered their expressed votes, and thus no way voters can help deter, detect, contain, and correct computer hacking in elections. That is, not only is it inappropriate to rely on voters to check whether BMDs alter expressed votes, *it doesn't work*.

Risk-limiting audits of a trustworthy paper trail can check whether errors in tabulating the votes *as recorded* altered election outcomes, but there is no way to check whether errors in how BMDs record *expressed* votes altered election outcomes. The outcomes of elections conducted on current BMDs therefore cannot be confirmed by audits. This paper identifies two properties of voting systems, *contestability* and *defensibility*, that are necessary conditions for any audit to confirm election outcomes. No commercially available EAC-certified BMD is contestable or defensible.

To reduce the risk that computers undetectably alter election results by printing erroneous votes on the official paper audit trail, the use of BMDs should be limited to voters who require assistive technology to vote independently.

Elections for public office and on public questions in the United States or any democracy must produce outcomes based on the votes that voters *express* when they indicate their choices on a paper ballot or on a machine. Computers have become indispensable to conducting elections, but computers are vulnerable. They can be hacked—compromised by insiders or external adversaries who can replace their software with fraudulent software that deliberately miscounts votes—and they can contain design errors and bugs—hardware or software flaws or configuration errors that result in mis-recording or mis-tabulating votes. Therefore there must be some way, *independent* of any software in any computers, to ensure that reported election outcomes are correct, i.e., consistent with the expressed votes as intended by the voters.

Voting systems should be *software independent*, meaning that “an undetected change or error in its software cannot cause an undetectable change or error in an election outcome” [23]. Indeed, version 2.0 of the Voluntary Voting System Guidelines (VVS 2.0) incorporates this principle [7].

Software independence is similar to tamper-evident packaging: if somebody opens the container and disturbs the contents, it will leave a trace.

While software independence is crucial, it is not enough: *who* can detect errors and *what happens* when errors are detected are just as important. Even if individual voters in principle could detect changes to their votes on the BMD-generated ballot, unless voters can provide convincing evidence of problems to the public and unless election officials take appropriate remedies when presented with such evidence, software independence alone does not guarantee that outcome-changing problems—accidental or malicious—can be caught, much less corrected.

To be acceptable, a voting system also must be *contestable*: We say that voting system is contestable if any change or error in its software that results in a change or error in a reported election outcome can generate public evidence that the reported outcome is not trustworthy. Evidence available only to individual voters¹ does not suffice: “trust me” is not evidence. If a voting system is contestable, it is software independent, but the converse is not necessarily true. If a voting system is not contestable, then problems might never see the light of day, much less be corrected.

Voting systems must also be *defensible*. We say that a voting system is defensible if, when it reports the correct outcome, it can also generate convincing public evidence that the reported outcome is correct. Evidence available only to an election official or voting system vendor does not suffice: in other words, “trust me” is not evidence. If a voting system is not defensible, then it is vulnerable to “crying wolf”: malicious actors could claim that the system malfunctioned when in fact it did not, and election officials will have no way to prove otherwise.

Rivest and Wack [23] also define a voting system to be *strongly software independent* if it is software independent and moreover, a detected change or error in an election outcome (due to change or error in the software) can be corrected using only the ballots and ballot records of the current election.² Strong software independence combines tamper evidence with a kind of resilience: there’s a way to tell whether faulty software caused a problem, and a way to recover from the problem if it did.

The only known practical technology for a contestable, defensible, strongly software independent voting system is *hand-marked paper ballots*, kept physically secure,

¹Specifically, if the voter is selected candidate A on the touchscreen of a BMD, but the BMD prints candidate B on the paper ballot, then this A-vs-B evidence is available to the individual voter, but the voter cannot demonstrate this evidence to anyone else, since nobody else saw—nor should have seen—where the voter touched the screen. Thus, the voting system cannot generate public evidence of errors recording expressed votes, even if those errors altered the reported outcome.

²The only alternative remedy would be to void the results of the entire election and conduct a new one.

counted by machine, audited manually, and recountable by hand.³

Over 40 states now use some form of paper ballot for most voters [14]. Most of the remaining states are taking steps to adopt paper ballots. But *not all voting systems that use paper ballots are equally secure*. Some are not software independent. Some are software independent but not contestable or defensible. In this report we explain:

- *Hand-marked paper ballot* systems are the only practical technology for contestable, defensible voting systems.
- *Some ballot-marking devices (BMDs)* can be software independent, but they are neither contestable nor defensible. Hacked or misprogrammed BMDs can alter election outcomes undetectably, and elections conducted using BMDs do not provide public evidence that reported outcomes are correct. Therefore BMDs should not be used by voters who are able to mark an optical-scan ballot with a pen.
- *All-in-one BMD or DRE+VVPAT voting machines* are not software independent, contestable, or defensible. They should not be used in public elections.

Terminology

Although a voter may form an intention to vote for a candidate or issue days, minutes, or seconds before actually casting a ballot, that intention is a psychological state that cannot be directly observed by anyone else. Others can have access to that intention through what the voter (privately) *expresses* to the voting technology by interacting with it, e.g., by making selections on a BMD or marking a ballot by hand.⁴ Voting systems must accurately record the vote as the voter *expressed* it.

With a *hand-marked paper ballot optical-scan* system, the voter is given a paper ballot on which all choices (candidates) in each contest are listed; next to each candidate

³The election must also generate convincing evidence that physical security of the ballots was not compromised, and the audit must generate convincing public evidence that the audit itself was conducted correctly.

⁴We recognize that voters make mistakes in expressing their intentions. For example, they may misunderstand the layout of a ballot or through a perceptual error or lapse of attention make an unintended choice. The use of touchscreen technology does not necessarily correct for such user errors, as every smartphone user who has mistyped an important text message knows. Poorly designed ballots, poorly designed touchscreen interfaces, and poorly designed assistive interfaces increase the rate of error in voters' expressions of their votes. For the purposes of this report, we assume that properly engineered systems seek to minimize such usability errors.

is a *target* (typically an oval or other shape) which the voter marks with a pen to indicate a vote. Ballots may be either preprinted or printed (unvoted) at the polling place using *ballot on demand* printers. In either case, the voter creates a tamper-evident record of intent by marking the printed paper ballot with a pen.

Such hand-marked paper ballots may be scanned and tabulated at the polling place using a *precinct-count optical scanner* (PCOS), or may be brought to a central place to be scanned and tabulated by a *central-count optical scanner* (CCOS). Mail-in ballots are typically counted by CCOS machines.

After scanning a ballot, a PCOS machine deposits the ballot in a secure, sealed ballot box for later use in recounts or audits; this is *ballot retention*. Ballots counted by CCOS are also retained for recounts or audits.⁵

Paper ballots can also be hand counted, but in most jurisdictions (especially where there are many contests on the ballot) this is hard to do quickly; Americans expect election-night reporting of unofficial totals. Hand counting—i.e., manually determining votes directly from the paper ballots—is appropriate for audits and recounts.

A *ballot-marking device* (BMD) provides a computerized user interface that presents the ballot to voters and captures their expressed selections, for instance, a touchscreen interface or an assistive interface that enables voters with disabilities to vote independently. Voter inputs (expressed votes) are recorded electronically. When a voter indicates that the ballot is complete and ready to be cast, the BMD prints a paper version of the electronically marked ballot. We generally use the term *BMD* for devices that mark ballots but do not tabulate or retain them, and *all-in-one* for devices that combine ballot marking, tabulation, and retention into the same paper path.

The paper ballot printed by a BMD may be in the same format as an optical-scan form (e.g., with ovals filled as if by hand) or it may list just the names of the candidate(s) selected in each contest. The BMD may also encode these selections into barcodes or QR codes for optical scanning. We discuss issues with barcodes later in this report.

An *all-in-one touchscreen voting machine* combines computerized ballot marking, tabulation, and retention in the same paper path. All-in-one machines come in several configurations:

- DRE+VVPAT machines—direct-recording electronic (DRE) voting machines with a voter-verifiable paper audit trail (VVPAT)—provide the voter a touchscreen (or

⁵Regulations and procedures governing custody and physical security of ballots are uneven and in many cases inadequate, but simple to correct because of decades of development of best practices.

other) interface, then print a paper ballot that is displayed to the voter under glass. The voter is expected to review this ballot and approve it, after which the machine deposits it into a ballot box. DRE+VVPAT machines do not contain optical scanners; that is, they do not read what is marked on the paper ballot; instead, they tabulate the vote directly from inputs to the touchscreen or other interface.

- BMD+Scanner all-in-one machines⁶ provide the voter a touchscreen (or other) interface to input ballot choices and print a paper ballot that is ejected from a slot for the voter to inspect. The voter then reinserts the ballot into the slot, after which the all-in-one BMD+scanner scans it and deposits it into a ballot box.

Opscan+BMD with separate paper paths. At least one model of voting machine (the Dominion ICP320) contains an optical scanner and a BMD in the same cabinet,⁷ so that the optical scanner and BMD-printer are not in the same paper path; no possible configuration of the software could cause a BMD-marked ballot to be deposited in the ballot box without human handling of the ballot. We do not classify this as an *all-in-one* machine.

Hacking

There are many forms of computer hacking. In this analysis of voting machines we focus on the alteration of voting machine software so that it miscounts votes or mis-marks ballots to alter election outcomes. There are many ways to alter the software of a voting machine: a person with physical access to the computer can open it and directly access the memory; one can plug in a special USB thumbdrive that exploits bugs and vulnerabilities in the computer's USB drivers; one can connect to its WiFi port or Bluetooth port or telephone modem (if any) and exploit bugs in those drivers, or in the operating system.

“Air-gapping” a system (which is to say, disconnecting it from a wired network) does not automatically protect it. Before each election, election administrators must transfer a *ballot definition* into the voting machine by inserting a *ballot definition cartridge* that was programmed on election-administration computers that may have been connected previously to various networks; it has been demonstrated that vote-changing viruses can propagate via these ballot-definition cartridges [13].

Hackers might be corrupt insiders with access to a voting-machine warehouse; cor-

⁶The ES&S ExpressVote can be configured as either a BMD or a BMD+Scanner all-in-one.

⁷More precisely, the ICP320 optical scanner and the BMD audio+buttons interface are in the same cabinet, but the printer is a separate box.

rupt insiders with access to a county's election-administration computers; outsiders who can gain remote access to election-administration computers; outsiders who can gain remote access to voting-machine manufacturers' computers (and "hack" the firmware installed in new machines, or the firmware updates supplied for existing machines), and so on. Supply-chain hacks are also possible: the hardware installed by a voting system vendor may have malware pre-installed by the vendor's component suppliers.⁸

Computer systems (including voting machines) have so many layers of software that it is impossible to make them perfectly secure [18, pp. 89–91]. When manufacturers of voting machines use the best known security practices, adversaries may find it more difficult to hack a BMD or optical scanner—but not impossible. Every computer in every critical system is vulnerable to compromise through hacking, insider attacks or exploiting design flaws.

Election assurance through risk-limiting audits.

To ensure that the reported outcome of each contest is that outcome that would have been found by accurately tabulating the voters' intent as recorded, the most practical known technology is a *risk-limiting audit* (RLA) of paper ballots [25, 26, 17]. The National Academies of Science, Engineering, and Medicine, recommend routine RLAs after every election [18], as do many other organizations and entities concerned with election integrity.⁹

A RLA involves manually inspecting randomly selected paper ballots following a rigorous protocol. The audit stops if and when the sample provides convincing evidence that the reported outcome is correct; otherwise, the audit continues until every ballot has been inspected manually and the correct electoral outcome is known.

RLAs can check whether errors in tabulating recorded votes altered election outcomes, but cannot check whether errors in recording expressed votes altered election outcomes. Properly preserved hand-marked paper ballots ensure that expressed votes are identical to recorded votes. On the other hand, BMDs might not record expressed

⁸Given that many chips and other components are manufactured in China and elsewhere, this is a serious concern. Carsten Schürmann has found Chinese pop songs on the internal memory of voting machines (C. Schürmann, personal communication, 2018). Presumably those files were left there accidentally—but this shows that malicious code *could* have been pre-installed deliberately, and that neither the vendor's nor the election official's security and quality control measures discovered and removed the extraneous files.

⁹Among them are the Presidential Commission on Election Administration, the American Statistical Association, the League of Women Voters, and Verified Voting Foundation.

votes accurately, for instance, if BMD software has bugs, was misconfigured, or was hacked. Thus, RLAs that rely on BMD output cannot ensure that election outcomes are correct.

RLAs protect against vote-tabulation errors, whether those errors are caused by failures to follow procedures, misconfiguration, miscalibration, faulty engineering, bugs, or malicious hacking.¹⁰ The *risk limit* of a risk-limiting audit is the maximum chance that an outcome that is incorrect because of tabulation errors will pass the audit without being corrected. The risk limit should be determined as a matter of policy or law. For instance, a 5% risk limit means that, if a reported outcome is wrong because of tabulation errors, there is at least a 95% chance that the post-election audit will correct it. Smaller risk limits give higher confidence in election outcomes, but require inspecting more ballots, other things being equal. RLAs never revise a correct outcome.

RLAs can be very efficient, depending in part on how the voting system is designed. If the computer results are accurate, an efficient RLA with a risk limit of 5% requires examining about (7 divided by the margin) ballots selected randomly from the contest.¹¹ For instance, if the margin of victory is 10% and the results are correct, the RLA would need to examine about $7/10\% = 70$ ballots to confirm the outcome at 5% risk. For a 1% margin, the RLA would need to examine about $7/1\% = 700$ ballots. The sample size does not depend (much) on the total number of ballot cast in the contest, only on the margin of the winning candidate's victory.

A paper-based voting system (such as one that uses optical scanners) is systematically more secure than a paperless system (such as DREs) only if the paper trail is trustworthy and the results are audited against the paper trail using a rigorous method such as an RLA.

But what if the paper ballots are not a trustworthy record of the votes expressed by the voters? If it is possible that error, hacking, bugs, or miscalibration caused the recorded votes to differ from the expressed votes, an RLA or even a full hand recount does not provide convincing public evidence that election outcomes are correct: such a system cannot be *defensible*. In short, paper ballots provide little assurance against hacking if they are never examined or if the paper might not accurately record the vote expressed by the voter.

¹⁰RLAs do not protect against problems that cause BMDs to print something other than what was shown to the voter on the screen, nor do they protect against problems with ballot custody.

¹¹Technically, it is the *diluted margin* that enters the calculation. The diluted margin is the number of votes that separate the winner with the fewest votes from the loser with the most votes, divided by the number of ballots cast, including undervotes and invalid votes.

Security Flaws

A BMD-generated paper trail is not a reliable record of the vote expressed by the voter. Like any computer, a BMD (or a DRE+VVPAT) is vulnerable to hacking, installation of unauthorized (fraudulent) software, and alteration of installed software.¹²

If a hacker sought to steal an election by altering BMD software, what would the hacker program the BMD to do? In cybersecurity practice, we call this the *threat model*.

The simplest threat model is this one: In some contests, not necessarily top-of-the-ticket, change a small percentage of the votes (such as 5%).

In recent national elections, analysts have considered a candidate who received 60% of the vote to have won by a landslide. Many contests are decided by less than a 10% margin. Changing 5% of the votes can change the margin by 10%, because “flipping” a vote for one candidate into a vote for a different candidate changes the difference in their tallies—i.e., the margin—by 2 votes. If hacking or bugs or misconfiguration could change 5% of the votes, that would be a very significant threat.

Although public and media interest often focus on top-of-the-ticket races such as President and Governor, elections for lower offices such as state representatives, who control legislative agendas and redistricting, and county officials, who manage elections and assess taxes, are just as important in our democracy. But most voters are not as familiar with the names of the candidates for those offices.

Research by one of us [9], in a real polling place in Tennessee during the 2018 election, found that half the voters *didn’t look at all* at the paper ballot printed by a BMD, even when they were holding it in their hand and directed to do so while carrying it from the BMD to the optical scanner. Those voters who did look at the BMD-printed ballot spent *an average of 4 seconds* examining it to verify that the eighteen or more choices they made were correctly recorded. That amounts to 222 milliseconds per contest, barely enough time for the human eye to move and refocus under perfect conditions and not nearly enough time for perception, comprehension, and recall [22].^{13 14}

¹²It is also vulnerable to bugs and misconfiguration.

¹³You might think, “the voter really *should* carefully review their BMD-printed ballot.” But because the scientific evidence shows that voters *do not* [9] and cognitively *cannot* [12] perform this task well, legislators and election administrators should provide a voting system that counts the votes *as voters express them*.

¹⁴Studies of voter confidence about their ability to verify their ballots are not relevant: in typical situations, subjective confidence and objective accuracy are at best weakly correlated. The relationship between confidence and accuracy has been studied in contexts ranging from eyewitness accuracy [6, 8,

The same study found that among voters who examined their hand-marked ballots, half were unable to recall key features of ballots cast moments before, a prerequisite step for being able to recall their own ballot choices.

Suppose, then, that 10% of voters examine their paper ballots carefully enough to even *see* the candidate's name recorded as their vote for legislator or county commissioner. Of those, perhaps only half will remember the name of the candidate they intended to vote for.¹⁵

Of those who notice that the vote printed is not the candidate they intended to vote for, what are they supposed to think, and what are they supposed to do? Do they think, "Oh, I must have made a mistake on the touchscreen," or do they think, "Hey, the machine is cheating or malfunctioning!" There's no way for the voter to know for sure—voters do make mistakes—and there's *absolutely* no way for the voter to prove to a pollworker or election official that a BMD printed something other than what the voter entered on the screen.¹⁶

Either way, polling place procedures generally advise voters to ask a pollworker for a new ballot if theirs does not show what they intended. Pollworkers should void that BMD-printed ballot, and the voter should get another chance to mark a ballot. Anecdotal evidence suggests that many voters are too timid to ask, or don't know that they have the right to ask, or are not sure whom to ask. Even if a voter asks for a new ballot, training for pollworkers is uneven, and we are aware of no formal procedure for resolving disputes if a request for a new ballot is refused. Moreover, there is no sensible protocol for ensuring that BMDs that misbehave are investigated—nor can there be, as we argue below.

Let's summarize. If a machine alters votes on 5% of the ballots (enabling it to change the margin by 10%), then optimistically we might expect $\frac{1}{20} \times \frac{1}{10} \times \frac{1}{2}$ or 0.25% of the voters to request a new ballot and correct their vote. This means that the machine will change the margin by 9.75% and get away with it.

In this scenario, 0.25% of the voters, one in every 400 voters, has requested a new ballot. You might think, "that's a form of *detection* of the hacking." But it isn't, as a

28] to confidence in psychological clinical assessments [10] and social predictions [11]. The disconnect is particularly severe at high confidence. Indeed, this is known as "the overconfidence effect." For a lay discussion, see *Thinking, Fast and Slow* by Nobel economist Daniel Kahnemann [15].

¹⁵We ask the reader, "do you know the name of the most recent losing candidate for county commissioner?" We recognize that some readers of this document *are* county commissioners, so we ask those readers to imagine the frame of mind of their constituents.

¹⁶Voters should *certainly* not videorecord themselves voting! That would defeat the privacy of the secret ballot and is illegal in most jurisdictions.

practical matter: a few individual voters may have detected that there was a problem, but there's no procedure by which this translates into any action that election administrators can take to correct the outcome of the election. Polling place procedures *cannot correct or deter hacking, or even reliably detect it*, as we discuss next. This is essentially the distinction between a system that is merely software independent and one that is contestable: a change to the software that alters the outcome might generate evidence for an alert, conscientious, individual voter, but it does not generate public evidence that an election official can rely on to conclude there is a problem.

Even if some voters notice that BMDs are altering votes, there's no way to correct the election outcome. Suppose a state election official wanted to detect whether the BMDs are cheating, and correct election results, based on actions by those few alert voters who notice the error. What procedures could possibly work against the manipulation we are considering?

1. How about, "If at least 1 in 400 voters claims that the machine misrepresented their vote, void the entire election."¹⁷ No responsible authority would implement such a procedure. A few dishonest voters could collaborate to invalidate entire elections simply by falsely claiming that BMDs changed their votes.
2. How about, "If at least 1 in 400 voters claims that the machine misrepresented their vote, then investigate." Investigations are fine, but then what? The only way an investigation can ensure that the outcome accurately reflects what voters expressed to the BMDs is to void an election in which the BMDs have altered votes and conduct a new election. But how do you know whether the BMDs have altered votes, except based the claims of the voters?¹⁸ Furthermore, the investigation itself would suffer from the same problem as above: how can one distinguish between voters who detected BMD hacking or bugs from voters who just want to interfere with an election?

This is the essential security flaw of BMDs: few voters will notice and promptly report discrepancies between what they saw on the screen and what is on the BMD print-

¹⁷Note that in many jurisdictions, far fewer than 400 voters use a given machine on election day: BMDs are typically expected to serve fewer than 300 voters per day. (The vendor ES&S recommended 27,000 BMDs to serve Georgia's 7 million voters, amounting to 260 voters per BMD [24].) Recall also that the rate 1 in 400 is tied to the amount of manipulation. What if the malware flipped only one vote in 50, instead of 1 vote in 20? That could still change the margin by 4%, but—in this hypothetical—would be noticed by only one voter in 1,000, rather than one in 400. The smaller the margin, the less manipulation it would have taken to alter the electoral outcome.

¹⁸Forensic examination of the BMD might show that it *was* hacked or misconfigured, but it cannot prove that the BMD *was not* hacked or misconfigured.

out, and even when they do notice, there's nothing appropriate that can be done. (Nor should it be the responsibility of voters to test voting-machine security and accuracy—this is a difficult burden that should not be placed on the voters.)

Therefore, BMDs should not be used by most voters.

Why can't we rely on pre-election and post-election logic and accuracy testing?

Most, if not all, jurisdictions perform some kind of *logic and accuracy testing* (LAT) of voting equipment before elections. LAT generally involves voting on the equipment using various combinations of selections, then checking whether the equipment tabulated the votes correctly. As the Volkswagen/Audi “Dieselgate” scandal shows, devices can be programmed to behave properly when they are tested but misbehave in use. Therefore, LAT can never prove that voting machines performed properly in practice.

Don't voters need to check hand-marked ballots, too? It is always a good idea to check one's work. The difference is, with hand-marked paper ballots, voters are responsible for catching and correcting *their own errors*, while if BMDs are used, voters are also responsible for catching *machine errors, bugs, and hacking*. Voters are the *only* people who can detect such problems with BMDs—but, as explained above, if voters do find problems, there's no way they can prove to poll workers or election officials that there were problems and no way to ensure that election officials take appropriate remedial action.

Other tradeoffs, BMDs versus hand-marked opscan

Supporters of ballot-marking devices advance several other arguments for their use.

- **Mark legibility.** A common argument is that a properly functioning BMD will generate clean, error-free, unambiguous marks, while hand marked paper ballots may contain mistakes and stray marks that make it impossible to discern a voter's intent. However appealing this argument seems at first blush, the data are not nearly so compelling. Experience with statewide recounts in Minnesota and elsewhere suggest that truly ambiguous handmade marks are very rare.¹⁹ For instance, 2.9 million hand-marked ballots were cast in the 2008 Minnesota race

¹⁹States do need clear and complete regulations for interpreting voter marks.

between Al Franken and Norm Coleman for the U.S. Senate. In a manual recount, between 99.95% and 99.99% of ballots were unambiguously marked.^{20 21} In addition, usability studies of hand marked bubble ballots—the kind in most common use in U.S. elections—indicate a *voter* error rate of 0.6%, much lower than the 2.5–3.7% error rate for machine-marked ballots [12].²² Moreover, modern image-based opscan equipment is better than older “marksense” machines at interpreting imperfect marks. Thus, mark legibility is not a good reason to adopt BMDs for all voters.

- **Undervotes, overvotes.** Another argument offered for BMDs is that the machines can alert voters to undervotes and prevent overvotes. That is true, but modern PCOS can also alert a voter to overvotes and undervotes, allowing a voter to eject the ballot and correct it. Other solutions, such as non-tabulating scanners that simply warn voters of overvotes and undervotes on hand-marked ballots, would be less risky than BMDs.
- **Bad ballot design.** Ill-designed paper ballots, just like ill-designed touchscreen interfaces, may lead to unintentional undervotes [19]. For instance, the 2006 Sarasota, Florida, touchscreen ballot was badly designed. The 2018 Broward County, Florida, opscan ballot was badly designed: it violated three separate guidelines from the EAC’s 2007 publication, “Effective Designs for the Administration of Federal Elections, Section 3: Optical scan ballots.” [27] In both of these cases (touchscreens in 2006, hand-marked optical-scan in 2018), undervote rates were high. The solution is to follow standard, published ballot-design guidelines and other best practices, both for touchscreens and for hand-marked ballots [3, 19].
- **Low-tech paper-ballot fraud.** All paper ballots, however they are marked, are vulnerable to *loss*, *ballot-box stuffing*, *alteration*, and *substitution* between the

²⁰ “During the recount, the Coleman and Franken campaigns initially challenged a total of 6,655 ballot-interpretation decisions made by the human recounters. The State Canvassing Board asked the campaigns to voluntarily withdraw all but their most serious challenges, and in the end approximately 1,325 challenges remained. That is, approximately 5 ballots in 10,000 were ambiguous enough that one side or the other felt like arguing about it. The State Canvassing Board, in the end, classified all but 248 of these ballots as votes for one candidate or another. That is, approximately 1 ballot in 10,000 was ambiguous enough that the bipartisan recount board could not determine an intent to vote.” [1] See also [20]

²¹ We have found that some local election officials consider marks to be ambiguous if *machines* cannot read the marks. That is a different issue from *humans* not being able to interpret the marks. Errors in machine interpretation of voter intent can be dealt with by manual audits: if the reported outcome is wrong because machines misinterpreted handmade marks, a RLA has a known, large chance of correcting the outcome.

²² Better designed user interfaces (UI) might reduce the error rate for machine-marked ballots below the historical rate for DREs; however, UI improvements cannot keep BMDs from printing something other than what the voter is shown on the screen.

time they are cast and the time they are recounted. That's why it is so important to make sure that ballot boxes are always in multiple-person (preferably bipartisan) custody at any time when they are handled, and that appropriate physical security measures are in place. Strong, verifiable chain-of-custody protections are essential.

Hand-marked paper ballots are vulnerable to alteration by anyone with a pen. Both hand-marked and BMD-marked paper ballots are vulnerable to substitution: anyone who has poorly supervised access to a legitimate BMD during election day can create fraudulent ballots, not necessarily to deposit them in the ballot box immediately (in case the ballot box is well supervised on election day) but with the hope of substituting it later in the chain of custody.²³

All those attacks (on hand-marked and on BMD-marked paper ballots) are fairly low-tech. There are also higher-tech ways of producing ballots indistinguishable from BMD-marked ballots for substitution into the ballot box, where there is inadequate chain-of-custody protection.

- **Accessible voting technology.** If everyone voted on a BMD, it would guarantee that an accessible device had been set up in the polling place for all voters who needed one. But this is not a good reason to adopt BMDs for *all* voters. Among other things, it would expose all voters to the security flaws described above, decreasing public confidence in the entire election. Some accessibility advocates argue that requiring disabled voters to use BMDs compromises their privacy since hand marked ballots are easily distinguishable from machine marked ballots. This argument has been undercut by the availability in the marketplace of BMDs that mark ballots that cannot easily be distinguished from hand marked ballots. Other advocates object to the idea that disabled voters must use a different method of marking ballots, arguing that their rights are thereby violated. Both HAVA and ADA require accommodations for voters with physical and cognitive impairments, but neither law requires that those accommodations must be used by all voters. To best enable and facilitate participation by all voters, each voter should be provided with a means of casting a vote best suited to their abilities.
- **Ballot printing costs.** Preprinted optical-scan ballots cost 20–50 cents each.²⁴ Blank cards for BMDs cost up to 15 cents each, depending on the make and model of BMD.²⁵ But optical-scan ballots must be preprinted for as many vot-

²³Some BMDs print a bar-code indicating when and where the ballot was produced, but that does not prevent such a substitution attack against currently EAC-certified, commercially available BMDs. We understand that systems under development might make ballot-substitution attacks against BMDs more difficult.

²⁴Single-sheet (one- or two-side) ballots cost 20-28 cents, double-sheet ballots needed for elections with many contests, up to 50 cents.

²⁵Ballot cards for ES&S ExpressVote cost about 15 cents. New Hampshire's (One4All / Prime III)

ers as *might* show up, whereas blank BMD cards are consumed in proportion to how many voters *do* show up. The Open Source Election Technology Institute (OSET) conducted an independent study of total life cycle costs²⁶ for hand-marked paper ballots and BMDs in conjunction with the 2019 Georgia legislative debate regarding BMDs [21]. OSET concluded that, even in the most optimistic (i.e., lowest cost) scenario for BMDs and the most pessimistic (i.e., highest cost) scenario for hand-marked paper ballots and ballot-on-demand (BOD) printers—which can print unmarked ballots as needed—the total lifecycle costs for BMDs would be higher than the corresponding costs for hand marked paper ballots.²⁷

- **Vote centers.** To run a vote center that serves many election districts with different ballot styles, one must be able to provide each voter a ballot containing the contests that voter is eligible to vote in, possibly in a number of different languages. This is easy with BMDs, which can be programmed with all the appropriate ballot definitions. With preprinted optical-scan ballots, the PCOS can be programmed to *accept* many different ballot styles, but the vote center must still maintain *inventory* of many different ballots. BOD printers are another economical alternative for vote centers.²⁸
- **Paper/storage.** BMDs that print summary cards rather than full-face ballots can save paper and storage space. However, many BMDs print full-face ballots, while many BMDs that print summary cards use thermal printers and paper that is fragile and can fade in a few months.²⁹

Advocates of hand-marked paper ballot systems advance these additional arguments.

BMDs used by sight-impaired voters use plain paper that is less expensive.

²⁶They include not only the cost of acquiring and implementing systems but also the ongoing licensing, logistics, and operating (purchasing paper stock, printing, and inventory management) costs.

²⁷BOD printers currently on the market arguably are best suited for vote centers, but less expensive options suited for polling places could be developed. Indeed, BMDs that print full-face ballots could be re-purposed as BOD printers for polling place use, with modest changes to the programming.

²⁸Ballot-on-demand printers *may* require maintenance such as replacement of toner cartridges. This is readily accomplished at a vote center with a professional staff. Ballot-on-demand printers may be a less attractive option for many small precincts on election day, where there is no professional staff—but on the other hand, they are less necessary, since far fewer ballot styles will be needed in any one precinct.

²⁹The California Top-To-Bottom Review (TTBR) of voting systems found that thermal paper can also be covertly spoiled wholesale using common household chemicals <https://votingsystems.cdn.sos.ca.gov/oversight/ttbr/red-diebold.pdf>, last visited 8 April 2019. The fact that thermal paper printing can fade or deteriorate rapidly might mean it does not satisfy the federal requirement to preserve voting materials for 22 months. <http://uscode.house.gov/view.xhtml?req=granuleid:USC-prelim-title52-section20701&num=0&edition=prelim>, last visited 8 April 2019.

- **Cost.** Using BMDs for all voters substantially increases the cost of acquiring, configuring, and maintaining the voting system. One PCOS can serve 1200 voters in a day, while one BMD can serve only about 260 [24]—though both these numbers vary greatly depending on the length of the ballot and the length of the day. OSET analyzed the relative costs of acquiring BMDs for Georgia’s nearly seven million registered voters versus a system of hand marked paper ballots, scanners, and BOD printers [21]. A BMD solution for Georgia would cost taxpayers between 3 and 5 times the cost of a system based on hand marked paper ballots.
- **Mechanical reliability and capacity.** Pens are likely to have less downtime than BMDs. It is easy and inexpensive to get more pens and privacy screens when additional capacity is needed. If a precinct-count scanner goes down, people can still mark ballots with a pen; if the BMD goes down, voting stops. Thermal printers used in DREs with VVPAT are prone to jams; those in BMDs might have similar flaws.

These secondary pros and cons of BMDs do not outweigh the primary security and accuracy concern: BMDs, if hacked or erroneous, can change votes in a way that is not correctable. BMD voting systems are not contestable, defensible, or strongly software independent. Therefore, ballots cast by BMD cannot effectively be audited.

Barcodes

A controversial feature of some BMDs allows them to print 1-dimensional or 2-dimensional barcodes on the paper ballots. A 1-dimensional barcode resembles the pattern of vertical lines used to identify products by their universal product codes. A 2-dimensional barcode or QR code is a rectangular area covered in coded image *modules* that encode more complex patterns and information. BMDs print barcodes on the same paper ballot that contains human-readable ballot choices. Voters using BMDs are expected to verify the human-readable printing on the paper ballot card, but the presence of barcodes with human-readable text poses some significant problems.

- **Barcodes are not human readable.** The whole purpose of a paper ballot is to be able to recount (or audit) the *voters’* votes in a way independent of any (possibly hacked or buggy) computers. If the official vote on the ballot card is the barcode, then it is impossible for the voters to verify that the official vote they cast is the vote they expressed. Therefore, before a state even *considers* using BMDs that print barcodes (and we do not recommend doing so), the State must ensure by statute that recounts and audits are based *only* on the human-readable portion of

the paper ballot. Even so, audits based on untrusted paper trails suffer from the verifiability the problems we outlined above.

- **Ballot cards with barcodes contain two different votes.** Suppose a state does ensure by statute that recounts and audits are based on the human-readable portion of the paper ballot. Now a BMD-marked ballot card with both barcodes and human-readable text contains two different votes in each contest: the barcode (used for electronic tabulation), and the human-readable selection printout (official for audits and recounts). In few (if any) states has there even been a discussion of the legal issues raised when the official markings to be counted differ between the original count and a recount.
- **Barcodes pose technical risks.** Any coded input into a computer system—including wired network packets, WiFi, USB thumbdrives, *and barcodes*—pose the risk that the input-processing software can be vulnerable to attack via deliberately ill-formed input. Over the past two decades, many such vulnerabilities have been documented on *each* of these channels (including barcode readers) that, in the worst case, give the attacker complete control of a system.³⁰ If an attacker were able to compromise a BMD, the barcodes are an attack vector for the attacker to take over an optical scanner (PCOS or CCOS), too. Since it is good practice to close down all such unneeded attack vectors into PCOS or CCOS voting machines (e.g., don’t connect your PCOS to the Internet!), it is also good practice to avoid unnecessary attack channels such as barcodes.

End-to-End Verifiable BMDs

In all BMD systems currently on the market, and in all BMD systems certified by the EAC, the printed ballot or ballot summary is the only channel by which voters can verify the correct recording of their ballots, independently of the computers. The analysis in this paper applies to all of those BMD systems.

There is a class of voting systems called “end-to-end verifiable” (E2E-V), which provide an alternate mechanism for voters to verify their votes [2]. Some E2E-V systems incorporate BMDs, for instance STAR-Vote³¹ [5]. If such a voting system could

³⁰An example of a barcode attack is based on the fact that many commercial barcode-scanner components (which system integrators use to build cash registers or voting machines) treat the barcode scanner using the same operating-system interface as if it were a keyboard device; and then some operating systems allow “keyboard escapes” or “keyboard function keys” to perform unexpected operations.

³¹The STAR-Vote system is actually a DRE+VVPAT system with a smart ballot box, rather than a BMD system: voters interact with a device that captures their votes electronically and prints a paper record that voters can inspect, but the electronic votes are held “in limbo” until the paper ballot is de-

be demonstrated to be contestable, defensible, and adequately usable by voters, then the analysis in this paper might not be applicable to such BMDs. No E2E-V systems are currently certified by the EAC, nor to our knowledge is any such system under review for certification, nor are any of the 5 major voting-machine vendors offering such a system for sale.³²

Design Flaws in All-in-One BMDs

Some voting machines incorporate a BMD interface, printer, and optical scanner into the same cabinet. Other DRE+VVPAT voting machines incorporate ballot-marking, tabulation, and paper-printout retention, but without scanning.

These are often called “all-in-one” voting machines. Any such machine that includes *ballot marking* and *deposit into the ballot box* in the same paper path, is unsafe.

Using an all-in-one machine, the voter makes choices on a touchscreen or through a different accessible interface. When the selections are complete, the BMD prints the completed ballot for the voter to review and verify, before depositing the ballot in a ballot box attached to the machine.

- The ES&S ExpressVote (in all-in-one mode) allows the voter to mark a ballot by touchscreen or audio interface, then prints a paper ballot card and ejects it from a slot. The voter has the opportunity to review the ballot, then the voter redeposits the ballot into the same slot, where it is scanned and deposited into a ballot box.
- The ES&S ExpressVoteXL allows the voter to mark a ballot by touchscreen or audio interface, then prints a paper ballot (cash-register tape format) and displays it under glass. The voter has the opportunity to review the ballot, then the voter touches the screen to indicate “OK,” and the machine pulls paper ballot up (still under glass) and into the integrated ballot box.
- The Dominion ImageCast Evolution (ICE) allows the voter to deposit a hand-marked paper ballot, which it scans and drops into the attached ballot box. *Or*, a voter can use a touchscreen or audio interface to direct the marking of a paper ballot, which the voting machine ejects through a slot for review; then the voter

posited in the smart ballot box. The ballot box does not read the votes from the ballot; rather, depositing the ballot tells the system that it has permission to cast the vote that it had already recorded from the touchscreen.

³²Some vendors, notably Scytl, have sold systems advertised as E2E-V in other countries. Those systems were not in fact E2E-V. Moreover, serious security flaws have been found in their implementations. See, e.g., [16].

redeposits the ballot into the slot, where it is scanned and dropped into the ballot box.

In all three of these machines, the ballot-marking printer is in the same paper path as the mechanism to deposit marked ballots into an attached ballot box. This opens up a very serious security vulnerability: the voting machine can mark the paper ballot (to add votes or spoil already-cast votes) after the last time the voter sees the paper, and then deposit that marked ballot into the ballot box without the possibility of detection.

Vote-stealing software could easily be constructed that looks for *undervotes* on the ballot, and marks those unvoted spaces for the candidate of the hacker's choice. This is very straightforward to do on optical-scan bubble ballots (as on the Dominion ICE) where undervotes are indicated by no mark at all. On machines such as the ExpressVote and ExpressVoteXL, the normal software indicates an undervote with the words NO SELECTION MADE on the ballot summary card. Hacked software could simply leave a blank space there (most voters wouldn't notice the difference), and then fill in that space and add a matching bar code after the voter has clicked "cast this ballot."

An even worse feature of the ES&S ExpressVote and the Dominion ICE is the *auto-cast* configuration setting (in the manufacturer's standard software) that allows the voter to indicate, "don't eject the ballot for my review, just print it and cast it without me looking at it." If fraudulent software were installed in the ExpressVote, it could change *all* the votes of any voter who selected this option, because the voting machine software would know *in advance of printing* that the voter had waived the opportunity to inspect the printed ballot. We call this auto-cast feature "permission to cheat" [4].

Regarding these all-in-one machines, we conclude:

- Any machine with ballot printing in the same paper path with ballot deposit is not *software independent*; it is *not* the case that "an error or fault in the voting system software or hardware cannot cause an undetectable change in election results." Therefore such all-in-one machines do not comply with the VVSG 2.0 (the Election Assistance Commission's Voluntary Voting Systems Guidelines).
- All-in-one machines on which all voters use the BMD interface to mark their ballots (such as the ExpressVote and ExpressVoteXL) *also* suffer from the same serious problem as ordinary BMDs: most voters do not review their ballots effectively, and elections on these machines are not contestable or defensible.
- The auto-cast option for a voter to allow the paper ballot to be cast without human inspection is particularly dangerous, and states must insist that vendors disable or eliminate this mode from the software. However, even disabling the auto-cast feature does not eliminate the risk of undetected vote manipulation.

Remark. The Dominion ImageCast Precinct ICP320 is a precinct-count optical scanner (PCOS) that also contains an audio+buttons ballot-marking interface for disabled voters. This machine can be configured to cast electronic-only ballots from the BMD interface, or an external printer can be attached to print paper optical-scan ballots from the BMD interface. When the external printer is used, that printer's paper path is *not* connected to the scanner+ballot-box paper path (a person must take the ballot from the printer and deposit it into the scanner slot). Therefore this machine is as safe to use as any PCOS with a separate external BMD.

Conclusion

Ballot-Marking Devices produce ballots that do not necessarily record the vote expressed by the voter when they enter their selections on the touchscreen: hacking, bugs, and configuration errors can cause the BMDs to print votes that differ from what the voter entered and verified electronically. Furthermore, in cases where the BMD-marked paper ballot does not record the expressed vote, the election system overall is not *contestable* or *defensible*, meaning that errors in elections conducted on compromised BMDs cannot be reliably detected or corrected, and that election officials cannot provide convincing evidence that correct reported outcomes of elections conducted using BMDs are indeed correct. Therefore BMDs should not be used by voters who can use hand-marked paper ballots.

All-in-one voting machines, that combine ballot-marking and ballot-box-deposit into the same paper path, are even worse. They have all the disadvantages of BMDs (they are neither contestable or defensible), and they can mark the ballot after the voter has inspected it. Therefore they are not even *software independent*, and should not be used by those voters who are capable of marking, handling, and visually inspecting a paper ballot.

When computers are used to record votes, the original transaction (the voter's expression of the votes) is not documented in a verifiable way.³³ When pen-and-paper is used to record the vote, the original expression of the vote *is* documented in a verifiable way (provided that secure chain of custody of paper ballots is maintained). Therefore, audits of elections conducted with BMDs cannot ensure that reported outcomes are correct, while audits of elections conducted with hand-marked paper ballots, counted by optical scanners, can.

³³It is conceivable that cryptographic protocols used in E2E-V systems could be used to create BMD-based systems that are contestable and defensible, but no such system exists, nor, to our knowledge, has such a design been worked out in principle.

References

- [1] A.W. Appel. Optical-scan voting extremely accurate in Minnesota. *Freedom to Tinker*, January 2009. <https://freedom-to-tinker.com/2009/01/21/optical-scan-voting-extremely-accurate-minnesota/>.
- [2] A.W. Appel. End-to-end verifiable elections. *Freedom to Tinker*, November 2018. <https://freedom-to-tinker.com/2018/11/05/end-to-end-verifiable-elections/>.
- [3] A.W. Appel. Florida is the Florida of ballot-design mistakes. *Freedom to Tinker*, November 2018. <https://freedom-to-tinker.com/2018/11/14/florida-is-the-florida-of-ballot-design-mistakes/>.
- [4] A.W. Appel. Serious design flaw in ESS ExpressVote touchscreen: “permission to cheat”. *Freedom to Tinker*, September 2018. <https://freedom-to-tinker.com/2018/09/14/serious-design-flaw-in-ess-expressvote-touchscreen-permission-to-cheat/>.
- [5] J. Benaloh, M. Byrne, B. Eakin, P. Kortum, N. McBurnett, O. Pereira, P.B. Stark, , and D.S. Wallach. Star-vote: A secure, transparent, auditable, and reliable voting system. *JETS: USENIX Journal of Election Technology and Systems*, 1:18–37, 2013.
- [6] R. K. Bothwell, K.A. Deffenbacher, and J.C. Brigham. Correlation of eyewitness accuracy and confidence: Optimality hypothesis revisited. *Journal of Applied Psychology*, 72:691–695, 1987.
- [7] Election Assistance Commission. Voluntary voting systems guidelines 2.0, September 2017. https://www.eac.gov/assets/1/6/TGDC_Recommended_VVSG2.0_P_Gs.pdf.
- [8] K. Deffenbacher. Eyewitness accuracy and confidence: Can we infer anything about their relation? *Law and Human Behavior*, 4:243–260, 1980.
- [9] R. DeMillo, R. Kadel, and M. Marks. What voters are asked to verify affects ballot verification: A quantitative analysis of voters’ memories of their ballots, November 2018. <https://ssrn.com/abstract=3292208>.
- [10] S.L. Desmarais, T.L. Nicholls, J. D. Read, and J. Brink. Confidence and accuracy in assessments of short-term risks presented by forensic psychiatric patients. *The Journal of Forensic Psychiatry & Psychology*, 21(1):1–22, 2010.

- [11] D. Dunning, D.W. Griffin, J.D. Milojkovic, and L. Ross. The overconfidence effect in social prediction. *Journal of Personality and Social Psychology*, 58:568–581, 1990.
- [12] S.P. Everett. *The Usability of Electronic Voting Machines and How Votes Can Be Changed Without Detection*. PhD thesis, Rice University, 2007.
- [13] A.J. Feldman, J.A. Halderman, and E.W. Felten. Security analysis of the Diebold AccuVote-TS voting machine. In *2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT 2007)*, August 2007.
- [14] Verified Voting Foundation. The verifier – polling place equipment – november 2018, November 2018. <https://www.verifiedvoting.org/verifier/>.
- [15] D. Kahnemann. *Thinking, fast and slow*. Farrar, Straus and Giroux, 2011.
- [16] S. J. Lewis, O. Pereira, and V. Teague. Ceci n’est pas une preuve: The use of trapdoor commitments in Bayer-Groth proofs and the implications for the verifiability of the Scytl-SwissPost Internet voting system, 2019. <https://people.eng.unimelb.edu.au/vjteague/UniversalVerifiabilitySwissPost.pdf>.
- [17] M. Lindeman and P.B. Stark. A gentle introduction to risk-limiting audits. *IEEE Security and Privacy*, 10:42–49, 2012.
- [18] National Academies of Sciences, Engineering, and Medicine. *Securing the Vote: Protecting American Democracy*. The National Academies Press, Washington, DC, September 2018.
- [19] L. Norden, M. Chen, D. Kimball, and W. Quesenbery. Better Ballots, 2008. Brennan Center for Justice, <http://www.brennancenter.org/publication/better-ballots>.
- [20] Office of the Minnesota Secretary of State. Minnesota’s historic 2008 election, 2009. <https://www.sos.state.mn.us/media/3078/minnesotas-historic-2008-election.pdf>.
- [21] E. Perez. Georgia state election technology acquisition: A reality check. OSET Institute Briefing, March 2019. https://trustthevote.org/wp-content/uploads/2019/03/06Mar19-OSETBriefing_GeorgiaSystemsCostAnalysis.pdf.

- [22] K. Rayner and M.S. Castelhana. Eye movements during reading, scene perception, and visual search, 2009. *Q J Experimental Psychology*, 2009, August 62(8), 1457-1506.
- [23] R.L. Rivest and J.P. Wack. On the notion of software independence in voting systems, July 2006. <http://vote.nist.gov/SI-in-voting.pdf>.
- [24] Election Systems and Software. State of Georgia Electronic Request for Information New Voting System Event Number: 47800-SOS0000035, 2018. <http://sos.ga.gov/admin/files/ESS%20RFI%20-%20Final%20-%20Redacted.pdf>.
- [25] P.B. Stark. Conservative statistical post-election audits. *Annals of Applied Statistics*, 2:550–581, 2008.
- [26] P.B. Stark. Risk-limiting post-election audits: P -values from common probability inequalities. *IEEE Transactions on Information Forensics and Security*, 4:1005–1014, 2009.
- [27] U. S. Election Assistance Commission. Effective designs for the administration of federal elections, June 2007. https://www.eac.gov/assets/1/1/EAC_Effective_Election_Design.pdf.
- [28] J.T. Wixted and G.L. Wells. The relationship between eyewitness confidence and identification accuracy: A new synthesis. *Psychological Science in the Public Interest*, 2017.

EXHIBIT C

Andrew W. Appel, Curriculum Vitae

Andrew W. Appel

Eugene Higgins Professor of Computer Science

[Department of Computer Science, Princeton University](#)

35 Olden Street, Princeton NJ 08540

appel@princeton.edu, +1-609-258-4627, fax: +1-609-258-2016

<https://www.cs.princeton.edu/~appel>

Research Interests

Software verification, programming languages, computer security, compilers, semantics, software engineering, information technology policy, elections and voting technology.

Education

A.B. *summa cum laude* ([physics](#)) [Princeton University](#), 1981

Ph.D. ([computer science](#)) [Carnegie-Mellon University](#), 1985

Professional Appointments

[Princeton University](#), Princeton, NJ. Eugene Higgins Professor of Computer Science, since 2011; Department Chair, 2009-15; Professor of Computer Science, since 1995; Associate Chair, 1997-2007; Assoc. Prof., 1992-95; Asst. Prof. 1986-92.

[Massachusetts Institute of Technology](#). Visiting Professor, July-December 2013.

[INRIA](#) (Institut National de Recherche en Informatique et en Automatique), Rocquencourt, France. Visiting Professor, academic year 2005-06 & summers 2004, 2007.

[Bell Laboratories](#), Murray Hill, NJ. Member of Technical Staff, Summer 1984. Consultant, 1983-2001.

[Carnegie-Mellon University](#), Pittsburgh, PA. Research and teaching assistant, 1982-85.

[College of Medicine](#), University of Illinois, Urbana, IL. Computer programmer, summers 1976-80.

Awards and Honors

Kusaka Memorial Prize in Physics, Princeton University, 1981.

National Science Foundation Graduate Student Fellowship, 1981-1984.

[ACM Fellow](#) ([Association for Computing Machinery](#)), 1998.

The Other Prize, Programming Contest of the ACM International Conference on Functional Programming, 1998.

[ACM SIGPLAN Distinguished Service Award](#), 2002.

ACM SIGPLAN selected "Real-time Concurrent Collection on Stock Multiprocessors" (Appel, Ellis, Li 1988) as one of the [50 most influential papers in 20 years of the PLDI conference](#), 2002.

Professional Activities

1. Program Committee, *ACM SIGPLAN '89 Conf. on Prog. Lang. Design and Implementation*, 1989.
2. Program Committee, *Seventeenth ACM Symp. on Principles of Programming Languages*, 1990.
3. Associate Editor, *ACM Transactions on Programming Languages and Systems*, 1990-1992.
4. Associate Editor, *ACM Letters on Programming Languages and Systems*, 1991-1992.
5. Program Chair, *Nineteenth ACM Symp. on Principles of Programming Languages*, 1992.
6. Co-editor, [Journal of Functional Programming](#) special issue on ML, 1992.
7. Program Committee, *Sixth ACM Conf. on Functional Prog. Lang. and Computer Architecture*, 1993.
8. Editor in Chief, *ACM Transactions on Programming Languages and Systems*, 1993-97.
9. Program Committee, *International Conference on Functional Programming*, 1997.
10. General Chair, [POPL'99: 26th ACM Symp. on Principles of Programming Languages](#), 1999.
11. Program Committee, *IEEE Symposium on Security and Privacy*, 2002.
12. Program Committee, *ACM SIGPLAN Workshop on Types in Language Design and Implementation*, 2003.
13. Program Committee, *Nineteenth Annual IEEE Symposium on Logic in Computer Science*, 2004.
14. Program Committee, *ACM SIGPLAN 2005 Conference on Programming Language Design and Implementation (PLDI)*, 2005.
15. Program Committee, [International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice \(LFMTP'06\)](#), 2006.
16. Program Committee, [EVT'07: 2007 Usenix/ACCURATE Electronic Voting Technology Workshop](#).
17. Program Committee, [POPL'09: 36th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages](#), 2009.
18. Program Committee, [PLDI 2011: 32nd ACM SIGPLAN conference on Programming Language Design and Implementation](#), 2011.
19. General Co-Chair, [ITP 2012: Interactive Theorem Proving](#), 2012.
20. Program Committee, [POPL 2014: 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages](#), 2014.
21. Award Committee, [SIGPLAN Programming Languages Software Award](#), 2016.
22. Board of Advisors, [Verified Voting Foundation](#), since 2015.
23. Program Committee, [POPL 2020: 47th ACM SIGPLAN Symposium on Principles of Programming Languages](#), 2020.

Research Grants

1. *Implementation of an efficient reducer for lambda expressions*, National Science Foundation DCR-8603453, \$115,799, 1986-88.
2. Digital Equipment Corporation Faculty Incentive Grant, \$180,000, 1986-89.
3. *Unifying compile-time and run-time evaluation*, National Science Foundation CCR-8806121, \$123,510, 1988-90.
4. *Standard ML of New Jersey software capitalization*, National Science Foundation CCR-8914570, \$119,545, 1990-91.
5. *Using immutable types for debugging and parallelism*, National Science Foundation CCR-9002786, \$174,618, 1990-92.
6. *Optimization of space usage*, National Science Foundation CCR-9200790, \$348,119, 1992-96.
7. *Framework, Algorithms, and Applications for Cross-module Inlining*, National Science Foundation CCR-9625413, \$180,331, 1996-98.
8. *Development of a HIL/LIL Framework for a National Compiler Infrastructure*, Defense Advanced Research Projects Agency and National Science Foundation (as subcontractor to Univ. of Virginia), \$1,397,293, 1996-99.

9. *Tools, Interfaces, and Access Control for Secure Programming*, National Science Foundation CCR-9870316, \$322,000, 1998-2001 (co-PI).
10. *Scaling Proof-Carrying Code to Production Compilers and Security Policies*, Defense Advanced Research Projects Agency, \$3,870,378, 1999-2004.
11. *Applying Compiler Techniques to Proof-Carrying Code*, National Science Foundation CCR-9974553, \$220,000, 1999-2002.
12. **IBM** University Partnership Program, \$40,000, 1999-2000.
13. *High-Assurance Common Language Runtime*, National Science Foundation CCR-0208601, \$400,000, 2002-2005.
14. *Assurance-Carrying Components*, Advanced Research and Development Agency contract NBCHC030106, \$759,910, 2003-05.
15. **Sun Microsystems** research grant, \$20,000, 2004.
16. *End-to-end source-to-object verification of interface safety*, National Science Foundation grant CCF-0540914, \$325,000, 2006-09.
17. *MulVAL Technologies Plan*, New Jersey Commission on Science and Technology, \$60,000, 2006.
18. Microsoft Corporation research grant, \$25,000, 2006.
19. *Evidence-based Trust in Large-scale MLS Systems*, Air Force Office of Scientific Research FA9550-09-1-0138 (as subcontractor to Kansas State University), \$1,000,000, 2009-14.
20. *Combining Foundational and Lightweight Formal Methods to Build Certifiably Dependable Software*, National Science Foundation grant CNS-0910448, \$500,000, 2009-13.
21. *CARS: A Platform for Scaling Formal Verification to Component-Based Vehicular Software Stacks*, Defense Advanced Research Projects Agency award FA8750-12-2-0293, \$6,108,346, 2012-2017.
22. *Verified HMAC*, Google Advanced Technology and Projects grant, \$95,928, 2014.
23. *Principled Optimizing Compilation of Dependently Typed Languages*, National Science Foundation grant CCF-1407794, \$600,000, 2014-17.
24. *Concurrent separation logic for C*, Intel Corporation research grant, \$238,015, 2015-16.
25. *Collaborative Research: Expeditions in Computing: The Science of Deep Specification*, National Science Foundation grant CCF-1521602, \$3,453,419, 2015-20.

Publications

Books, chapters in books



1. "Garbage Collection," in *Topics in Advanced Language Implementation*, Peter Lee, ed. MIT Press, 1991.
2. *Compiling with Continuations*, Cambridge University Press, 1992.
3. *Modern Compiler Implementation in ML*, Cambridge University Press, 1998.
4. *Modern Compiler Implementation in Java*, Cambridge University Press, 1998.
5. *Modern Compiler Implementation in C*, Cambridge University Press, 1998.
6. *Modern Compiler Implementation in Java, 2nd edition*, with Jens Palsberg, Cambridge University Press, 2002.
7. *Alan Turing's Systems of Logic: The Princeton Thesis*, edited and introduced by Andrew W. Appel, Princeton University Press, 2012.
8. *Program Logics for Certified Compilers*, by Andrew W. Appel with Robert Dockins, Aquinas Hobor, Lennart Beringer, Josiah Dodds, Gordon Stewart, Sandrine Blazy, and Xavier Leroy. Cambridge University Press, 2014.
9. *Verified Functional Algorithms*, by Andrew W. Appel, 2017. Volume 3 of *Software Foundations*, edited by B. C. Pierce.

Journal papers, refereed conference papers, and patents

10. A Microprocessor-Based CAI System with Graphic Capabilities, by Frank J. Mabry, Allan H. Levy, and Andrew W. Appel, *Proc. 1978 conference, Assoc. for Development of Computer-based Instruction Systems*.
11. *Rogomatic: A Belligerent Expert System*, by Michael L. Mauldin, Guy J. Jacobson, Andrew W. Appel, and Leonard G. C. Hamey. *Proc. Fifth Nat. Conf. Canadian Soc. for Computational Studies of Intelligence*, May 1984.
12. An Efficient Program for Many-Body Simulations. *SIAM Journal on Scientific and Statistical Computing* 6(1):85-103, 1985.
13. *Semantics-Directed Code Generation*, by Andrew W. Appel, *Proc. Twelfth ACM Symposium on Principles of Programming Languages*, January 1985.
14. *Generalizations of the Sethi-Ullman algorithm for register allocation*. Andrew W. Appel and Kenneth J. Supowit, *Software Practice and Experience* 17(6):417-421, 1987.
15. *A Standard ML compiler*, by Andrew W. Appel and David B. MacQueen, *Proc. Third Int'l Conf. on Functional Programming & Computer Architecture (LNCS 274, Springer-Verlag)*, Portland, Oregon, September 1987.
16. *Garbage collection can be faster than stack allocation*. Andrew W. Appel. *Information Processing Letters* 25(4):275-279, 17 June 1987.
17. *Real-time concurrent collection on stock multiprocessors*, by Andrew W. Appel, John Ellis, and Kai Li, *Proc. ACM SIGPLAN '88 Conf. on Prog. Lang. Design & Implementation*, pp. 11-20, June 1988.
18. *The World's Fastest Scrabble Program*. Andrew W. Appel and Guy J. Jacobson, *Comm. ACM* 31(5):572-578, May 1988.
19. *Simulating digital circuits with one bit per wire*. Andrew W. Appel, *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems* 7(9):987-993, September 1988.
20. *Continuation-passing, closure-passing style*, by Andrew W. Appel and Trevor Jim, *Proc. Sixteenth ACM Symposium on Principles of Programming Languages*, pp. 293-302, January 1989.
21. *Simple Generational Garbage Collection and Fast Allocation*. Andrew W. Appel. *Software--Practice and Experience* 19(2):171-183, February 1989.
22. *Allocation without Locking*. Andrew W. Appel. *Software--Practice and Experience* 19(7):703-705, July 1989.
23. *Runtime Tags Aren't Necessary*. Andrew W. Appel. *Lisp and Symbolic Computation* 2, 153-162 (1989).
24. *Vectorized Garbage Collection*. Andrew W. Appel and Aage Bendiksen. *The Journal of Supercomputing* 3, 151-160 (1989).
25. *A Runtime System*. *Lisp and Symbolic Computation* 3, 343-380, 1990.
26. *An advisor for flexible working sets*, by Rafael Alonso and Andrew W. Appel, *1990 ACM SIGMETRICS Conf. on Measurement and Modeling of Computer Systems*, pp. 153-162, May 1990.
27. *Debugging Standard ML without reverse engineering*, by Andrew P. Tolmach and Andrew W. Appel, *Proc. 1990 ACM Conf. on Lisp and Functional Programming*, pp. 1-12, June 1990.

28. [Real-time concurrent garbage collection system and method](#), by John R. Ellis, Kai Li, and Andrew W. Appel. U.S. Patent 5,088,036, 1992.
29. [Virtual memory primitives for user programs](#), by Andrew W. Appel and Kai Li, *Proc. Fourth Int'l Conf. on Architectural Support for Prog. Languages and Operating Systems*, (SIGPLAN Notices 26(4)) pp. 96-107, April 1991.
30. [Standard ML of New Jersey](#), by Andrew W. Appel and David B. MacQueen, *Third Int'l Symp. on Prog. Lang. Implementation and Logic Programming*, Springer-Verlag LNCS 528, pp. 1-13, August 1991.
31. [Callee-save registers in Continuation-Passing Style](#), by Andrew W. Appel and Zhong Shao. *Lisp and Symbolic Computation* 5, 189-219, 1992.
32. [Smartest Recompile](#), by Zhong Shao and Andrew W. Appel, *Proc. Twentieth ACM Symp. on Principles of Programming Languages*, January 1993.
33. [A Critique of Standard ML](#). Andrew W. Appel. *Journal of Functional Programming* 3 (4) 391-430, 1993.
34. [Unrolling Lists](#), by Zhong Shao, John H. Reppy, and Andrew W. Appel, *Proc. 1994 ACM Conf. on Lisp and Functional Programming*, pp. 185-195, June 1994.
35. [Space-Efficient Closure Representations](#), by Zhong Shao and Andrew W. Appel, *Proc. 1994 ACM Conf. on Lisp and Functional Programming*, pp. 150-161, June 1994.
36. [Separate Compilation for Standard ML](#), by Andrew W. Appel and David B. MacQueen, *Proc. 1994 ACM Conf. on Programming Language Design and Implementation* (SIGPLAN Notices v. 29 #6), pp. 13-23, June 1994.
37. [Axiomatic Bootstrapping: A guide for compiler hackers](#), Andrew W. Appel, *ACM Transactions on Programming Languages and Systems*, vol. 16, number 6, pp. 1699-1718, November 1994.
38. [Loop Headers in Lambda-calculus or CPS](#). Andrew W. Appel. *Lisp and Symbolic Computation* 7, 337-343, 1994.
39. [A Debugger for Standard ML](#). Andrew Tolmach and Andrew W. Appel. *Journal of Functional Programming*, vol. 5, number 2, pp. 155-200, April 1995.
40. [A Type-Based Compiler for Standard ML](#), by Zhong Shao and Andrew W. Appel, *Proc. 1995 ACM Conf. on Programming Language Design and Implementation* (SIGPLAN Notices v. 30 #6), pp. 116-129, June 1995.
41. [Cache Performance of Fast-Allocating Programs](#), by Marcelo J. R. Goncalves and Andrew W. Appel, *Proc. Seventh Int'l Conf. on Functional Programming and Computer Architecture*, pp. 293-305, ACM Press, June 1995.
42. [Empirical and Analytic Study of Stack versus Heap Cost for Languages with Closures](#). Andrew W. Appel and Zhong Shao. *Journal of Functional Programming* 6 (1) 47-74, 1996.
43. [How to Edit a Journal by E-mail](#). Andrew W. Appel *Journal of Scholarly Publishing* 27 (2) 82-99, January 1996.
44. [Iterated Register Coalescing](#), by Lal George and Andrew W. Appel, *23rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* pp. 208-218, January 1996.
45. [Iterated Register Coalescing](#). Lal George and Andrew W. Appel. *ACM Transactions on Programming Languages and Systems* 18(3) 300-324, May 1996. [Shorter version](#) appeared in *23rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, January 1996.
46. [Security and document compatibility for electronic refereeing](#). Andrew W. Appel. *CBE Views* 20(1), 1997, published by the Council of Biology Editors.
47. [Lambda-Splitting: A Higher-Order Approach to Cross-Module Optimizations](#), by Matthias Blume and Andrew W. Appel, *Proc. ACM SIGPLAN International Conference on Functional Programming (ICFP '97)*, pp. 112-124, June 1997.
48. [The Zephyr Abstract Syntax Description Language](#), by Daniel C. Wang, Andrew W. Appel, Jeff L. Korn, and Christopher S. Serra. *Conference on Domain-Specific Languages*, USENIX Association, October 1997.
49. [Shrinking Lambda Expressions in Linear Time](#). Andrew W. Appel and Trevor Jim. *Journal of Functional Programming* v. 7 no. 5, pp. 515-540, 1997.
50. [Traversal-based Visualization of Data Structures](#), by Jeffrey L. Korn and Andrew W. Appel, *IEEE Symposium on Information Visualization (InfoVis '98)*, pp. 11-18, October 1998.
51. [Hierarchical Modularity](#). Matthias Blume and Andrew W. Appel, *ACM Transactions on Programming Languages and Systems*, 21 (4) 812-846, July 1999.

52. [Lightweight Lemmas in Lambda Prolog](#), by Andrew W. Appel and Amy Felty, *16th International Conference on Logic Programming*, pp. 411-425, MIT Press, November 1999.
53. [Proof-Carrying Authentication](#), by Andrew W. Appel and Edward Felten, *6th ACM Conference on Computer and Communications Security*, November 1999.
54. [Efficient and Safe-for-Space Closure Conversion](#), Zhong Shao and Andrew W. Appel, *ACM Trans. on Prog. Lang. and Systems* 22(1) 129-161, January 2000.
55. [A Semantic Model of Types and Machine Instructions for Proof-Carrying Code](#), by Andrew W. Appel and Amy P. Felty. *27th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '00)*, pp. 243-253, January 2000.
56. [Machine Instruction Syntax and Semantics in Higher Order Logic](#), by Neophytos G. Michael and Andrew W. Appel, *17th International Conference on Automated Deduction (CADE-17)*, Springer-Verlag (Lecture Notes in Artificial Intelligence), pp. 7-24, June 2000.
57. [Technological Access Control Interferes with Noninfringing Scholarship](#). Andrew W. Appel and Edward W. Felten. *Communications of the ACM* 43 (9) 21-23, September 2000.
58. [An Indexed Model of Recursive Types for Foundational Proof-Carrying Code](#). Andrew W. Appel and David McAllester. *ACM Transactions on Programming Languages and Systems* 23 (5) 657-683, September 2001.
59. [Type-Preserving Garbage Collectors](#), Daniel C. Wang and Andrew W. Appel, *POPL 2001: The 28th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 166-178, January 2001.
60. [SAFKASI: A Security Mechanism for Language-Based Systems](#), Dan S. Wallach, Andrew W. Appel, and Edward W. Felten. *ACM Transactions on Software Engineering and Methodology*, 9 (4) 341-378, October 2000.
61. [Optimal Spilling for CISC Machines with Few Registers](#), by Andrew W. Appel and Lal George. *ACM SIGPLAN 2001 Conference on Programming Language Design and Implementation*, pp. 243-253, June 2001.
62. [Foundational Proof-Carrying Code](#), by Andrew W. Appel, *16th Annual IEEE Symposium on Logic in Computer Science (LICS '01)*, pp. 247-258, June 2001.
63. [A Stratified Semantics of General References Embeddable in Higher-Order Logic](#), by Amal Ahmed, Andrew W. Appel, and Roberto Virga. *17th Annual IEEE Symposium on Logic in Computer Science (LICS 2002)*, pp. 75-86, June 2002.
64. [Creating and Preserving Locality of Java Applications at Allocation and Garbage Collection Times](#), by Yefim Shuf, Manish Gupta, Hubertus Franke, Andrew W. Appel, and Jaswinder Pal Singh. *17th Annual ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA 2002)*, *SIGPLAN Notices* 37(11) pp. 13-25, November 2002.
65. [Mechanisms for secure modular programming in Java](#), by Lujo Bauer, Andrew W. Appel, and Edward W. Felten. *Software--Practice and Experience* 33:461-480, 2003.
66. [A Trustworthy Proof Checker](#), by Andrew W. Appel, Neophytos G. Michael, Aaron Stump, and Roberto Virga. *Journal of Automated Reasoning* 31:231-260, 2003.
67. [Using Memory Errors to Attack a Virtual Machine](#), by Sudhakar Govindavajhala and Andrew W. Appel, *2003 IEEE Symposium on Security and Privacy*, pp. 154-165, May 2003.
68. [A Provably Sound TAL for Back-end Optimization](#), by Juan Chen, Dinghao Wu, Andrew W. Appel, and Hai Fang. *PLDI 2003: ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 208-219, June 2003.
69. [Foundational Proof Checkers with Small Witnesses](#), by Dinghao Wu, Andrew W. Appel, and Aaron Stump. *5th ACM-SIGPLAN International Conference on Principles and Practice of Declarative Programming*, pp. 264-274, August 2003.
70. [Policy-Enforced Linking of Untrusted Components \(Extended Abstract\)](#), by Eunyoung Lee and Andrew W. Appel, *European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pp. 371-374, September 2003.
71. [Polymorphic Lemmas and Definitions in Lambda Prolog and Twelf](#), by Andrew W. Appel and Amy P. Felty. *Theory and Practice of Logic Programming* 4 (1) 1-39, January 2004.
72. [Dependent Types Ensure Partial Correctness of Theorem Provers](#), by Andrew W. Appel and Amy P. Felty. *Journal of Functional Programming* 14(1):3-19, January 2004.

73. [Construction of a Semantic Model for a Typed Assembly Language](#), by Gang Tan, Andrew W. Appel, Kedar N. Swadi, and Dinghao Wu. In *5th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI '04)*, January 2004.
74. [MulVAL: A Logic-based Network Security Analyzer](#) by Xinming Ou, Sudhakar Govindavajhala, and Andrew W. Appel, In *14th Usenix Security Symposium*, August 2005.
75. [A Compositional Logic for Control Flow](#) by Gang Tan and Andrew W. Appel, in *7th International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI)*, January 2006.
76. [Safe Java Native Interface](#), by Gang Tan, Andrew W. Appel, Srimat Chakradhar, Anand Raghunathan, Srivaths Ravi, and Daniel Wang. *International Symposium on Secure Software Engineering*, March 2006.
77. [A Very Modal Model of a Modern, Major, General Type System](#), by Andrew W. Appel, Paul-Andre Mellies, Christopher D. Richards, and Jerome Vouillon. *POPL 2007: The 34th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, January 2007.
78. [Separation Logic for Small-step C minor](#), by Andrew W. Appel and Sandrine Blazy, in *TPHOLs 2007: 20th International Conference on Theorem Proving in Higher-Order Logics*, pp. 5-21, September 2007.
79. [Oracle Semantics for Concurrent Separation Logic](#), by Aquinas Hobor, Andrew W. Appel, and Francesco Zappa Nardelli, in *ESOP'08: European Symposium on Programming*, April 2008.
80. [Multimodal Separation Logic for Reasoning About Operational Semantics](#), by Robert Dockins, Andrew W. Appel, and Aquinas Hobor, in *Twenty-fourth Conference on the Mathematical Foundations of Programming Semantics*, May 2008.
81. [The New Jersey Voting-machine Lawsuit and the AVC Advantage DRE Voting Machine](#), by Andrew W. Appel, Maia Ginsburg, Harri Hursti, Brian W. Kernighan, Christopher D. Richards, Gang Tan, and Penny Venetis. In *EVT/WOTE'09, 2009 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections*, August 2009.
82. [A Fresh Look at Separation Algebras and Share Accounting](#) by Robert Dockins, Aquinas Hobor, and Andrew W. Appel. *Seventh Asian Symposium on Programming Languages and Systems (APLAS 2009)*, December 2009.
83. [A Theory of Indirection via Approximation](#), by Aquinas Hobor, Robert Dockins, and Andrew W. Appel. *POPL 2010: The 37th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 171-184, January 2010.
84. [Formal Verification of Coalescing Graph-Coloring Register Allocation](#), by Sandrine Blazy, Benoit Robillard and Andrew W. Appel. *ESOP 2010: 19th European Symposium on Programming*, pp. 145-164, March 2010.
85. [Concurrent Separation Logic for Pipelined Parallelization](#), by Christian J. Bell, Andrew W. Appel, and David Walker. In *SAS 2010: 17th Annual Static Analysis Symposium*, September 2010.
86. [Semantic Foundations for Typed Assembly Languages](#), by A. Ahmed, A. W. Appel, C. D. Richards, K. Swadi, G. Tan, and D. C. Wang. *ACM Transactions on Programming Languages and Systems*, 32(3):7.1-7.67, March 2010.
87. [A Logical Mix of Approximation and Separation](#) by Aquinas Hobor, Robert Dockins, and Andrew W. Appel. In *APLAS 2010: 8th ASIAN Symposium on Programming Languages and Systems*, November 2010.
88. [Local Actions for a Curry-style Operational Semantics](#) by Gordon Stewart and Andrew W. Appel. In *PLPV'11: 5th ACM SIGPLAN Workshop on Programming Languages meets Program Verification*, January 29, 2011.
89. [Verified Software Toolchain](#), by Andrew W. Appel. In *ESOP 2011: 20th European Symposium on Programming*, LNCS 6602, pp. 1-17, March 2011.
90. [VeriSmall: Verified Smallfoot Shape Analysis](#), by Andrew W. Appel. In *CPP 2011: First International Conference on Certified Programs and Proofs*, Springer LNCS 7086, pp. 231-246, December 2011.
91. [A Certificate Infrastructure for Machine-Checked Proofs of Conditional Information Flow](#), by Torben Amtoft, Josiah Dodds, Zhi Zhang, Andrew Appel, Lennart Beringer, John Hatcliff, Xinming Ou and Andrew Cousino. *First Conference on Principles of Security and Trust (POST 2012)*, LNCS 7215, pp. 369-389, March 2012.
92. [A list-machine benchmark for mechanized metatheory](#) by Andrew W. Appel, Robert Dockins, and Xavier Leroy. *Journal of Automated Reasoning* 49(3):453-491, 2012. DOI 10.1007/s10817-011-9226-1

93. [Security Seals On Voting Machines: A Case Study](#), by Andrew W. Appel. *ACM Transactions on Information and System Security (TISSEC)* 14 (2) pages 18:1--18:29, September 2011.
94. [Verified Heap Theorem Prover by Paramodulation](#), by Gordon Stewart, Lennart Beringer, and Andrew W. Appel. In *ICFP 2012: The 17th ACM SIGPLAN International Conference on Functional Programming*, pp. 3-14, September 2012.
95. [Mostly Sound Type System Improves a Foundational Program Verifier](#), by Josiah Dodds and Andrew W. Appel. *3rd International Conference on Certified Programs and Proofs (CPP 2013)*, December 2013.
96. [Verified Compilation for Shared-memory C](#), by Lennart Beringer, Gordon Stewart, Robert Dockins, and Andrew W. Appel. *ESOP'14: 23rd European Symposium on Programming*, April 2014.
97. [Portable Software Fault Isolation](#), by Joshua A. Kroll, Gordon Stewart, and Andrew W. Appel. *CSF'14: Computer Security Foundations Symposium*, IEEE Press, July 2014.
98. [Compositional CompCert](#), by Gordon Stewart, Lennart Beringer, Santiago Cuellar, and Andrew W. Appel. *POPL 2015: The 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 275-287, January 2015.
99. [Verified Correctness and Security of OpenSSL HMAC](#), by Lennart Beringer, Adam Petcher, Katherine Q. Ye, and Andrew W. Appel. In *24th USENIX Security Symposium*, pages 207-221, August 2015.
100. [Verification of a Cryptographic Primitive: SHA-256](#), by Andrew W. Appel. *ACM Transactions on Programming Languages and Systems*, 37(2) 7:1-7:31, April 2015.
101. [Modular Verification for Computer Security](#), by Andrew W. Appel, in *29th IEEE Computer Security Foundations Symposium (CSF'16)*, June 2016.
102. [Shrink Fast Correctly!](#) by Olivier Savary Belanger and Andrew W. Appel. *Proceedings of International Symposium on Principles and Practice of Declarative Programming (PPDP'17)*, 12 pages, October 2017 (PPDP'17).
103. [Verified Correctness and Security of mbedTLS HMAC-DRBG](#) by Katherine Q. Ye, Matthew Green, Naphat Sanguansin, Lennart Beringer, Adam Petcher, and Andrew W. Appel. *CCS'17: ACM Conference on Computer and Communications Security*, October 2017.
104. [Bringing order to the separation logic jungle](#), by Qinxiang Cao, Santiago Cuellar, and Andrew W. Appel. *APLAS'17: 15th Asian Symposium on Programming Languages and Systems*, November 2017.
105. [A verified messaging system](#), by William Mansky, Andrew W. Appel, and Aleksey Nogin. *OOPSLA'17: ACM Conference on Object-Oriented Programming Systems, Languages, and Applications*, October 2017. *Proceedings of the ACM on Programming Languages (PACM/PL)* volume 1, issue OOPSLA, paper 87, 2017.
106. [Position paper: the science of deep specification](#), by Andrew W. Appel, Lennart Beringer, Adam Chlipala, Benjamin C. Pierce, Zhong Shao, Stephanie Weirich and Steve Zdancewic, *Philosophical Transactions of the Royal Society A* 375:21060331 (24 pages), 2017.
107. [VST-Floyd: A separation logic tool to verify correctness of C programs](#), by Qinxiang Cao, Lennart Beringer, Samuel Gruetter, Josiah Dodds, and Andrew W. Appel. *Journal of Automated Reasoning* 61(1), pp. 367-422, 2018. (Local copy)
108. [Closure Conversion is Safe for Space](#), by Zoe Paraskevopoulou and Andrew W. Appel. *Proceedings of the ACM on Programming Languages*, vol. 3, no. ICFP, article 83, 29 pages, doi 10.1145/3341687, August 2019.
109. [Abstraction and Subsumption in Modular Verification of C Programs](#), by Lennart Beringer and Andrew W. Appel. *FM2019: 23rd International Symposium on Formal Methods*, October 2019.
110. [Connecting Higher-Order Separation Logic to a First-Order Outside World](#), by William Mansky, Wolf Honoré, and Andrew W. Appel, *ESOP 2020: European Symposium on Programming*, April 2020.
111. [Ballot-Marking Devices \(BMDs\) Cannot Assure the Will of the Voters](#), by Andrew W. Appel, Richard A. DeMillo, and Philip B. Stark. To appear in *Election Law Journal*, 2020. (Earlier versions [appeared on SSRN](#).)
112. [Verified sequential malloc/free](#), by Andrew W. Appel and David A. Naumann, in *2020 ACM SIGPLAN International Symposium on Memory Management*, June 2020.

Workshop and refereed conference papers

113. [Debuggable concurrency extensions for Standard ML](#), by Andrew P. Tolmach and Andrew W. Appel, *Proc. ACM/ONR Workshop on Parallel and Distributed Debugging*, May 1991 (SIGPLAN Notices, Dec.

- 1991), pp. 115-127.
114. [Efficient Substitution in Hoare Logic Expressions](#), by Andrew W. Appel, Kedar Swadi, and Roberto Virga. *4th International Workshop on Higher-Order Operational Techniques in Semantics (HOOTS 2000)*, pp. 35-50, September 2000.
 115. [Fair use, public domain, or piracy ... should the digital exchange of copyrighted works be permitted or prevented? \(Rountable Panel II: Digital Video\)](#), by Andrew W. Appel, Jeffrey Cunard, Martin Garbus, and Edward Hernstadt, *Fordham Intellectual Property, Media & Entertainment Law Journal*, volume 11, number 2, page 317, 2001.
 116. [A Trustworthy Proof Checker](#), by Andrew W. Appel, Neophytos G. Michael, Aaron Stump, and Roberto Virga. In *Verification Workshop - VERIFY 2002* and (jointly) in *Foundations of Computer Security - FCS 2002* Copenhagen, Denmark, July 25-26, 2002.
 117. [A list-machine benchmark for mechanized metatheory \(extended abstract\)](#) by Andrew W. Appel and Xavier Leroy. *LFMTP'06: International Workshop on Logical Frameworks and Meta-Languages: Theory and Practice*, August 2006.
 118. [Effective Audit Policy for Voter-Verified Paper Ballots](#), presented at *2007 Annual Meeting of the American Political Science Association*, Chicago, September 1, 2007.

Review Articles, Tutorials, Position Papers

119. [Book Review of *Garbage Collection: Algorithms for Automatic Dynamic Memory Management*](#) by Richard Jones and Rafael Lins. *Journal of Functional Programming* 7(2), pp. 227-229, March 1997.
120. [SSA is Functional Programming](#). *ACM SIGPLAN Notices* v. 33, no. 4, pp. 17-20, April 1998.
121. [Protection against untrusted code](#). *IBM Developer Works*, September 1999.
122. Retrospective: Real-time Concurrent Collection on Stock Multiprocessors. *20 Years of the ACM/SIGPLAN Conference on Programming Language Design and Implementation (1979-1999): A Selection*, ACM Press, 2004.
123. [Foundational High-level Static Analysis](#). In *CAV 2008 Workshop on Exploiting Concurrency Efficiently and Correctly*, July 2008.
124. [Technical Perspective: The Scalability of CertiKOS](#), by Andrew W. Appel, *Communications of the ACM*, vol. 62 no.10, page 88. DOI [10.1145/3356906](https://doi.org/10.1145/3356906).
125. [Freedom-to-Tinker](#): 16 [articles on the freedom-to-tinker.com blog](#) between 2007 and 2009; 6 articles in 2010; 15 articles in 2011.
126. [The Birth of Computer Science at Princeton in the 1930s](#), in A. W. Appel, ed., *Alan Turing's Systems of Logic: The Princeton Thesis*, Princeton University Press, 2012.
127. [Research Needs for Secure, Trustworthy, and Reliable Semiconductors](#), by Andrew Appel, Chris Daverse, Kenneth Hines, Rafic Makki, Keith Marzullo, Celia Merzbacher, Ron Perez, Fred Schneider, Mani Soma, and Yervant Zorian. Final workshop report of the NSF/CCC/SRC workshop on Convergence of Software Assurance Methodologies and Trustworthy Semiconductor Design and Manufacture, 2013.
128. [CertiCoq: A verified compiler for Coq](#), by Abhishek Anand, Andrew Appel, Greg Morrisett, Zoe Paraskevopoulou, Randy Pollack, Olivier Savary Belanger, Matthieu Sozeau, and Matthew Weaver. In *CoqPL'17: The Third International Workshop on Coq for Programming Languages*, January 2017.
129. [Position paper: the science of deep specification](#), by Andrew W. Appel, Lennart Beringer, Adam Chlipala, Benjamin C. Pierce, Zhong Shao, Stephanie Weirich, Steve Zdancewic. *Philosophical Transactions of the Royal Society A* vol. 375, no. 2104, September 2017.
130. [Securing the Vote: Protecting American Democracy](#), by National Academies of Science, Engineering, and Medicine: Lee C. Bollinger, Michael A. McRobbie, Andrew W. Appel, Josh Benaloh, Karen Cook, Dana DeBeauvoir, Moon Duchin, Juan E. Gilbert, Susan L. Graham, Neal Kelley, Kevin J. Kennedy, Nathaniel Persily, Ronald L. Rivest, Charles Stewart III. September 2018.
131. [Evidence-Based Elections: Create a Meaningful Paper Trail, then Audit](#), by Andrew W. Appel and Philip B. Stark, *Georgetown Law Technology Review*, volume 4, pages 523-541, 2020.

Unrefereed papers

132. [An Investigation of Galaxy Clustering Using an Asymptotically Fast N-Body Algorithm](#). Senior Thesis, Princeton University, 1981.
133. [Compile-time Evaluation and Code Generation in Semantics-Directed Compilers](#). Ph.D. Thesis, Carnegie-Mellon University, July 1985.
134. [Concise specifications of locally optimal code generators](#), Princeton Univ. Dept. of Computer Science CS-TR-080-87, 1987.
135. [Re-opening closures](#), Princeton Univ. Dept. of Computer Science CS-TR-079-87, February 1987.
136. [Optimizing closure environment representations](#), by Andrew W. Appel and Trevor Jim. Princeton Univ. Dept. of Computer Science CS-TR-168-88, July 1988.
137. [Unifying Exceptions with Constructors in Standard ML](#), with David MacQueen, Robin Milner, and Mads Tofte. Univ. of Edinburgh Dept. of Comp. Sci. CSR-266-88, May 1988.
138. [Profiling in the presence of optimization and garbage collection](#), by Andrew W. Appel, Bruce Duba, and David MacQueen. CS-TR-197-88, November 1988.
139. [Hash-Consing Garbage Collection](#), by Andrew W. Appel and Marcelo J.R. Goncalves, Technical report TR-412-93, Department of Computer Science, Princeton University, January 1993.
140. [Emulating Write-Allocate on a No-Write-Allocate Cache](#), by Andrew W. Appel, CS-TR-459-94, Princeton University, June 20, 1994.
141. [Is POPL Mathematics or Science?](#), by Andrew W. Appel, *ACM SIGPLAN Notices* 27 (4), pp. 87-89, April 1992.
142. [Intensional Equality ;=\) for Continuations](#), by Andrew W. Appel, *ACM SIGPLAN Notices* 31 (2), pp. 55-57, February 1996.
143. [Ceci n'est pas une urne: On the Internet vote for the Assemblée des Français de l'Etranger](#), by Andrew W. Appel, June 2006.
144. [Insecurities and Inaccuracies of the Sequoia AVC Advantage 9.00H DRE Voting Machine](#), by Andrew W. Appel, Maia Ginsburg, Harri Hursti, Brian W. Kernighan, Christopher D. Richards, and Gang Tan. October 2008.
145. [The CompCert Memory Model, Version 2](#), by Xavier Leroy, Andrew W. Appel, Sandrine Blazy, and Gordon Stewart. INRIA Research Report RR-7987, June 2012.
146. [Compiler Correctness for Concurrency: from concurrent separation logic to shared-memory assembly language](#), by Santiago Cuellar, Nick Giannarakis, Jean-Marie Madiot, William Mansky, Lennart Beringer, and Andrew W. Appel, Technical report TR-014-19, Department of Computer Science, Princeton University, March 2020.
147. [Fair Elections During a Crisis: Urgent Recommendations in Law, Media, Politics, and Tech to Advance the Legitimacy of, and the Public Confidence in, the November 2020 U.S. Elections.](#), by the Ad Hoc Committee for 2020 Election Fairness and Legitimacy (Appel, Azari, Cain, *et al.*), edited by Richard L. Hasen, UCI Law School, April 2020.

PhD Students

1. [Andrew P. Tolmach](#), Ph.D. (1992) [Debugging Standard ML](#). Professor, Portland State University.
2. [Zhong Shao](#), Ph.D. (1994) [Compiling Standard ML for Efficient Execution on Modern Machines](#). Professor, Yale University.
3. [Marcelo J. R. Goncalves](#), Ph.D. (1995) [Cache Performance of Programs with Intensive Heap Allocation and Generational Garbage Collection](#).
4. [Matthias Blume](#), Ph.D. (1997) [Hierarchical Modularity and Intermodule Optimization](#). Computer Scientist, Google, Inc.
5. [Richard \(Drew\) Dean](#), Ph.D. (1999) [Formal Aspects of Mobile Code Security](#). Senior Computer Scientist, SRI International.
6. [Jeffrey L. Korn](#), Ph.D. (1999) [Abstraction and Visualization in Graphical Debuggers](#). Software Engineer, Google, Inc.
7. [Daniel C. Wang](#), Ph.D. (2002) [Managing Memory with Types](#). Computer Scientist, Amazon.com.
8. [Kedar N. Swadi](#), Ph.D. (2003) [Typed Machine Language](#). CTO, AlgoAnalytics, Pune, India.

9. [Lujo Bauer](#), Ph.D. (2003) *Access Control for the Web via Proof-Carrying Authorization*. Associate Professor, Carnegie Mellon University.
10. [Eunyoung Lee](#), Ph.D. (2003) *Secure Linking: A Logical Framework for Policy-Enforced Component Composition*. Associate Professor, Dongduk Women's University, Seoul, Korea.
11. [Juan Chen](#), Ph.D. (2004) *A Low-Level Typed Assembly Language with a Machine-checkable Soundness Proof*. Computer Scientist, Google, Inc.
12. [Amal J. Ahmed](#), Ph.D. (2004) *Semantics of Types for Mutable State*. Associate Professor, Northeastern University.
13. [Gang Tan](#), Ph.D. (2005) *A Compositional Logic for Control Flow and its Application to Foundational Proof-Carrying Code*. Professor, Pennsylvania State University.
14. [Dinghao Wu](#), Ph.D. (2005) *Interfacing Compilers, Proof Checkers, and Proofs for Foundational Proof-Carrying Code*. Associate Professor, Pennsylvania State University.
15. [Xinming Ou](#), Ph.D. (2005) *A Logic Programming Approach to Network Security Analysis*. Professor, University of South Florida.
16. [Sudhakar Govindavajhala](#), Ph.D. (2006) *A Formal Approach to Practical Network Security Management*. Computer and network security consultant.
17. [Aquinas Hobor](#), Ph.D. (2008) *Oracle Semantics*. Assistant Professor, National University of Singapore and Yale/NUS college.
18. [Christopher D. Richards](#), Ph.D. (2010) *The Approximation Modality in Models of Higher-Order Types*. Computer Scientist, Google, Inc.
19. [Robert Dockins](#), Ph.D. (2012) *Operational Refinement for Compiler Correctness*. Researcher, Galois.com.
20. [James Gordon Stewart](#), Ph.D. (2015) *Verified Separate Compilation for C*. Assistant Professor, Ohio University.
21. [Josiah Dodds](#), Ph.D. (2015) *Computation Improves Interactive Symbolic Execution*. Researcher, Galois.com.
22. [Qinxiang Cao](#), Ph.D. (2018) *Separation-Logic-based Program Verification in Coq*. Assistant Professor, Shanghai Jiao Tong University.
23. [Olivier Savary Bélanger](#), Ph.D. (2019) *Verified Extraction for Coq*. Researcher, Galois.com.
24. [Santiago Cuellar](#), PhD (2020) *Concurrent Permission Machine for modular proofs of optimizing compilers with shared memory concurrency*. Researcher, Galois.com.

The documents linked from this page are included to ensure timely dissemination of scholarly and technical work on a non-commercial basis. Copyright and all rights therein are maintained by the authors or by other copyright holders, notwithstanding that they have offered their works here electronically. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each author's copyright. These works may not be reposted without the explicit permission of the copyright holder.